



0400
09.10.01

500.40572X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): N. SAGAWA.
Serial No.: 09 / 942,215
Filed: AUGUST 30, 2001
Title: TRANSACTION PROCESSING SYSTEM HAVING SERVICE
LEVEL CONTROL CAPABILITIES.

#2

LETTER CLAIMING RIGHT OF PRIORITY

Assistant Commissioner for
Patents
Washington, D.C. 20231

SEPTEMBER 20, 2001

Sir:

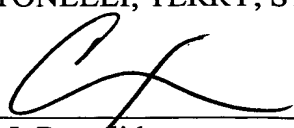
Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s)
the right of priority based on:

Japanese Patent Application No. 2001 - 028231
Filed: FEBRUARY 5, 2001

A certified copy of said Japanese Patent Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP



Carl I. Brundidge
Registration No. 29,621

CIB/rp
Attachment



日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2001年 2月 5日

出願番号

Application Number:

特願2001-028231

出願人

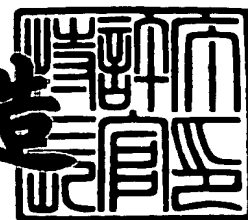
Applicant(s):

株式会社日立製作所

2001年 8月31日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造



出証番号 出証特2001-3077453

【書類名】 特許願

【整理番号】 GM0011039

【提出日】 平成13年 2月 5日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/00

【発明者】

 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所 中央研究所内

 【氏名】 佐川 暢俊

【特許出願人】

 【識別番号】 000005108

 【氏名又は名称】 株式会社日立製作所

【代理人】

 【識別番号】 100075513

 【弁理士】

 【氏名又は名称】 後藤 政喜

【選任した代理人】

 【識別番号】 100084537

 【弁理士】

 【氏名又は名称】 松田 嘉夫

【選任した代理人】

 【識別番号】 100114236

 【弁理士】

 【氏名又は名称】 藤井 正弘

【手数料の表示】

 【予納台帳番号】 019839

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

【物件名】	図面	1
【物件名】	要約書	1
【プルーフの要否】	要	

【書類名】 明細書

【発明の名称】 サービスレベル制御機構を有するトランザクション処理システム及びそのためのプログラム

【特許請求の範囲】

【請求項 1】

1 つまたは複数のサービスを提供し、該提供する各サービスには 1 つまたは複数のクライアントが接続可能なトランザクション処理システムであって、

前記提供するサービスに対応して定義された優先度情報を保持する優先度情報保持手段と、

前記優先度情報保持手段に保持された優先度情報を参照して前記クライアントからトランザクション処理システムに投入された処理要求の実行優先度を決定する実行優先度決定手段と、を有し、

該決定された実行優先度にしたがって処理を行うことを特徴とするトランザクション処理システム。

【請求項 2】

前記実行優先度決定手段は、提供するサービスに対応して定義された優先度情報に高い優先度が指定されたサービスに対する処理要求に対して、高い実行優先度を与えること特徴とする請求項 1 に記載のトランザクション処理システム。

【請求項 3】

1 つまたは複数のサービスを提供し、該提供する各サービスには 1 つまたは複数のクライアントが接続可能なトランザクション処理システムであって、

前記提供するサービスに対応して定義された優先度情報を保持する優先度情報保持手段と、

前記提供するサービスに対してクライアントから送付された処理要求を、前記提供するサービス毎に順序付けて格納するキューイング手段と、

格納した処理要求の滞留情報を該キューイング手段から取得する滞留情報取得手段と、

前記優先度情報保持手段に保持された優先度情報及び前記取得された滞留情報を参照して前記クライアントからトランザクション処理システムに投入された処

理要求の実行優先度を決定する実行優先度決定手段と、を有し、

該決定された実行優先度にしたがって処理を行うことを特徴とするトランザクション処理システム。

【請求項 4】

前記滞留情報取得手段は、

前記キューイング手段中に滞留している処理要求の数と、

前記キューイング手段中に滞留している処理要求の到着時刻と、

を取得することを特徴とする請求項 3 に記載のトランザクション処理システム。

【請求項 5】

前記実行優先度決定手段は、

前記提供するサービスに対応して定義された許容待ち時間と、前記処理要求の滞留情報取得手段により取得された処理要求の到着時刻を比較することにより優先度を決定することを特徴とする請求項 4 に記載のトランザクション処理システム。

【請求項 6】

1 つまたは複数のサービスを提供し、該提供する各サービスには 1 つまたは複数のクライアントが接続可能であり、該提供する各サービスは 1 つまたは複数の実行モジュールより構成されるトランザクション処理システムであって、

前記各サービスを構成する 1 つまたは複数の実行モジュールの識別子を格納する手段と、

前記実行モジュールを記憶する記憶手段と、

該各実行モジュールの更新を前記識別子に基づいて管理する更新管理手段と、を有し、

該更新管理手段により該実行モジュールが更新された場合には、前記サービスに対応するトランザクションの開始に先立って、更新された実行モジュールを記憶手段へ配置することを特徴とするトランザクション処理システム。

【請求項 7】

前記更新管理手段は、前記各サービス毎に、1 つまたは複数の実行モジュールの更新と、実行モジュールの更新の検出とを排他的に実行することを特徴とする

請求項 6 に記載のトランザクション処理システム。

【請求項 8】

1 つまたは複数のサービスを提供し、該提供する各サービスには 1 つまたは複数のクライアントが接続可能なトランザクション処理システムであって、

該クライアントから前記提供するサービスに対して送付された処理要求を提供するサービス毎に順序付けて格納するキューイング手段と、

該キューイング手段に格納した処理要求の滞留情報を取得する滞留情報取得手段と、

前記各サービスに対するトランザクションのスループットを検出するスループット検出手段と、

前記各サービスに対してトランザクションの処理プロセスを割り当てるプロセス割り当て手段と、を有し、

該プロセス割り当て手段は、取得した処理要求滞留情報と検出したトランザクションスループットとを参照し、前記各サービスに対してプロセスの割り当てを決定することを特徴とするトランザクション処理システム。

【請求項 9】

前記プロセス割り当て手段は、

前記キューイング手段に格納された処理要求が増加する場合には、割り当てるプロセス数を増加する一方、キューイング手段に格納された処理要求が減少する場合には、割り当てるプロセス数を減少することを特徴とする請求項 8 に記載のトランザクション処理システム。

【請求項 10】

前記プロセス割り当て手段は、前記各サービスに応じて付与された優先度順にプロセスの割り当てを行うことを特徴とする請求項 8 に記載のトランザクション処理システム。

【請求項 11】

1 つまたは複数のサービスを提供し、該提供する各サービスには 1 つまたは複数のクライアントが接続可能なトランザクション処理をコンピュータに実行させるプログラムであって、

前記提供するサービスに対応して定義された優先度情報を優先度情報データベースへ格納する手段と、

前記クライアントから提供するサービスに対して送付された処理要求を、前記提供するサービス毎に順序付けてキューへ格納する手段と、

該キューに格納した処理要求の滞留情報を取得する手段と、

前記優先度情報および前記滞留情報を参照してトランザクション処理に投入された前記クライアントからの処理要求の実行優先度を決定する手段と

該決定された実行優先度にしたがってトランザクションの処理を行う手段とをコンピュータに機能させることを特徴とするプログラム。

【請求項 1 2】

1 つまたは複数のサービスを提供し、該提供する各サービスには 1 つまたは複数のクライアントが接続可能であり、該提供する各サービスは 1 つまたは複数の実行モジュールより構成されるトランザクション処理をコンピュータに実行させるプログラムであって、

前記各サービスを構成する 1 つまたは複数の実行モジュールの識別子に基づいて各実行モジュールの更新状況を判定する手段と、

前記実行モジュールの更新が実行された場合には、サービスに対応するトランザクションの開始に先立って、更新された実行モジュールを記憶手段へ配置する手段とをコンピュータに機能させることを特徴とするプログラム。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、トランザクション処理システムに関し、特に、複数の顧客からのトランザクション処理を行うものに関する。

【0 0 0 2】

【従来の技術】

トランザクション処理システムは、主に金融取引や受発注処理など企業情報システムの基幹系において、多数の処理要求を効率よく、かつ整合性を保証しながら実行するためのシステムである。通常はクライアント・サーバ型のシステム構

成がとられ、クライアント（端末）が要求を発行し、サーバが必要に応じてデータベースにアクセスしながら業務処理本体を実行する。サーバにて実際の業務を実行する処理プログラムをサービスと呼ぶ。

【0003】

また、トランザクション処理システムにおいてサービスを提供する側をサービスプロバイダと呼ぶ。例えば、銀行リテール業務では、ATMやテラーがクライアントであり、顧客口座データベースを有する基幹システムがサーバとなる。そして、銀行がサービスプロバイダであり、預金の引き出し、預け入れ業務などはサービスに相当する。

【0004】

サーバ側で用いられるトランザクション処理ミドルウェアが、トランザクションモニタである。その役割は、主に以下の2点である。

【0005】

(1) クライアントから送られる処理要求を受け付けてキューイングし、要求の優先度やサーバの混雑度を勘案しながら、適当なサーバプログラム（サービス）に順次制御を渡す。これにより、サーバ資源の有効活用を図る。

【0006】

(2) 処理中に生起するエラーや障害を検出し、処理が成功裡に完結した場合には結果の書き込み処理（コミット）を、完結しなかった場合にはキャンセル（ロールバック）や再起動（リラン）処理を行う。これにより、トランザクション処理の整合性を保証する。

【0007】

ここで、トランザクションモニタの一般的な構成を図18に示す。

【0008】

トランザクションモニタ209はサーバ215上に位置し、処理要求を発行するクライアントプログラムはクライアント端末221、222上に位置する。

【0009】

サーバ215は通常UNIXサーバ、メインフレームコンピュータなどであり、クライアント端末はパーソナルコンピュータ、ATM端末などである。

【0010】

トランザクションモニタ209は、リクエストキュー210、211と、スケジューラ204、およびトランザクション実行部207を有する。

【0011】

サービスに対する処理要求は一般にメッセージ（電文）の形でクライアント221、222からサーバ215に渡される。このため、トランザクションモニタ209は通信処理機能を有し、これを制御しながら処理メッセージを受け付ける。

【0012】

受信されたメッセージは、トランザクションモニタ209内に処理要求（リクエスト）として格納される。通常複数のリクエストがトランザクションモニタ209内に滞留することになるので、リクエストは先入れ先出しのデータ構造であるキュー210、211を用いて入力された順番に格納される。格納されたリクエストは、トランザクション実行部207の資源（CPU、メモリなど）が空き次第、格納した順にキュー201、211から取り出され、対応するサービスプログラム206によって処理される。

【0013】

（スケジューリングと負荷分散）

リクエストをキューから取り出しサービスプログラムによる処理に移すことを、スケジューリングという。スケジューリングを効率よく行うことは、トランザクション処理システムの効率を上げるために重要である。

【0014】

特にサービスを提供する資源（プロセッサ、サーバなど）が複数存在する場合、各資源に対しどのようにリクエストを割り振るかが処理効率を左右する。複数資源に対してリクエストを適切に割り当てトランザクションの処理効率を上げることを、負荷分散と呼ぶ。以下では、上記スケジューリングと負荷分散を含め、リクエストと資源との割り当て処理全般をスケジューリングと呼ぶことがある。

【0015】

スケジューリングの方式としては、サービスを提供するプロセス数の増減によ

る負荷分散が知られている。図 1 9 を用いてその概要を説明する。

【 0 0 1 6 】

図 1 9 において、リクエストキュー 3 0 0 には、リクエスト 3 0 1 ~ 3 0 9 が格納されており、これらはプロセス 3 1 0 ~ 3 1 3 によって順次処理される。ここでプロセスとはプログラムが計算機上で処理される単位であり、一つの仮想アドレス空間と、そこにローディングされたプログラム、データ、およびプログラム実行状態を示す CPU レジスタの組よりなる。

【 0 0 1 7 】

図 1 9 の例では、各プロセスには同一のトランザクション処理プログラム（サービスプログラム）がローディングされている。計算機の CPU に空きがある間は、プロセス数を増やすことによって同時に実行されるサービス数を増やし、CPU の利用率を向上させることができる。

【 0 0 1 8 】

したがって、トランザクションに割り当てるプロセス数を増減することで、トランザクションの処理スループット（単位時間に処理されるリクエスト数）の調整が可能となる。

【 0 0 1 9 】

図 1 9 （ a ）は、キュー 3 0 0 にたまったリクエスト数が少ない状況を示す。トランザクションモニタは、リクエスト数に応じて、本サービスに少ないプロセス数（ 3 1 0 、 3 1 1 ）を割り当てる。

【 0 0 2 0 】

図 1 9 （ b ）は、多数のメッセージが到着し、キュー 3 0 0 の待ち合わせリクエスト数が増加した状況を示す。トランザクションモニタはキューの状況を監視し、当該サービスに割り当てるプロセス数を増やす（ 3 1 0 ~ 3 1 3 ）。

【 0 0 2 1 】

図 1 9 （ c ）は、到着するメッセージが減少し、キューが短くなった状況である。トランザクションモニタは空いたプロセス 3 1 3 を当該サービスから除き、他のサービスまたは仕事に割り当てる。以上のように、キューの長さで割り当てるプロセス数を関連付けることによって、CPU リソースの範囲内でトランザク

ションの効率を向上させることができる。

【 0 0 2 2 】

また、トランザクションモニタの制御するサーバが複数存在する場合には、サーバ間の負荷分散として図 2 0 に示す方式が用いられる。

【 0 0 2 3 】

今、サーバが 3 台 (4 2 0 ~ 4 2 2) 存在し、そのうち 1 台 (サーバ 4 2 0) のキュー 4 0 0 が、サーバの処理能力や混雑度などの問題で他サーバのキュー 4 0 1、4 0 2 に比べて長くなったとする。クライアント側の処理プログラム 4 3 1 はこの状態を検出し、キューの短いサーバ 4 2 1、4 2 2 に対してメッセージを優先的に送付するように制御する。これにより、複数のサーバ間でのキューの長さが平均化され、総合的なスループットを向上させることができる。

【 0 0 2 4 】

(メッセージブローカ)

トランザクションモニタを応用してさらに高度な複合サービスを実現するシステムとして、メッセージブローカがある。

【 0 0 2 5 】

通常のトランザクション処理システムでは、一つのメッセージに対して一つのサービスが対応するのに対し、メッセージブローカでは、一つのメッセージを複数のサービス間で引き継ぎ、連鎖的に呼び出して処理を行う。あるメッセージに対し、どのサービスをどの順序で呼び出すかは、サービスフロー (業務フロー) としてトランザクションモニタ内に指定する。呼び出されるサービス自身は、トランザクションモニタと同じサーバにあっても良いし、他の独立なサーバにあっても良い。

【 0 0 2 6 】

メッセージブローカの構成を図 2 1 に示す。

【 0 0 2 7 】

処理要求を発行するクライアントプログラム 5 0 1、5 0 2 はクライアント端末上に位置し、トランザクションモニタ 5 0 9 は、トランザクション処理用のサーバに位置する。

【 0 0 2 8 】

業務サービスを提供するサービスプログラム A 5 3 0、サービスプログラム B 5 3 1 は、それぞれ別個のサーバ（あるいは同一のサーバでも良い）にローディングされる。端末および各サーバの間は、メッセージの通信路が確保されている。トランザクションモニタ 5 0 9 は、リクエストキュー 5 1 0、5 1 1 と、リクエストの処理順番を決定するスケジューラ 5 0 4 を有する。

【 0 0 2 9 】

通常のトランザクション処理システム（上記図 1 8）と比較して、メッセージブローカではトランザクション実行部（図 1 8 の 2 0 7）の機能が拡張され、サービスフロー実行部 5 0 7 を構成する。

【 0 0 3 0 】

サービスフロー実行部 5 0 7 においては、メッセージに応じて単なるサービスプログラムを起動、実行するのではなく、サービスプロバイダによって定義されたサービスフローの実行を管理する。

【 0 0 3 1 】

このようにメッセージブローカでは、トランザクションモニタ上でサービスフローの実行を可能とすることにより、複数のサービスプログラムを組み合わせてより複雑なサービスを構築することができる。

【 0 0 3 2 】

（稼働中のノード入れ替え）

メッセージブローカでは、業務の更新や追加によってサービスフローが頻繁に変更される可能性がある。サービスフローの変更のたびにシステム全体を停止させることは望ましくないので、システムを稼働させつつサービスフローのみを変更する仕組みが必要となる。

【 0 0 3 3 】

一つの方法は、サービスフローを実行するプロセス群を 2 つのグループ（現用グループと待機グループ）に分け、現用グループで変更前のフローを実行しつつ、待機グループ上に新しいサービスフローの再ローディングを行うことである。ローディングが終了した段階でメッセージのルーティングを現用グループから待

機グループへ切り替えて運用を続行する。

【0034】

もう一つの方法は、サービスノード毎にルーティング可・不可の状態を設け、入れ替え対象ノードのルーティングを不可にし、ノードへのメッセージ入力を抑止した上でノードの再ローディングを行うことである。

【0035】

上記のようなトランザクションまたはメッセージブローカの処理システムとしては、特開平9-62624号公報（オンライントランザクション処理システム）、特開平6-243077号公報（分散型トランザクション処理システム）、特開平8-63432号公報（優先順位を考慮したトランザクション一括処理システム）、特開平6-52121号公報（バッチ処理、リアルタイム処理振り分け型トランザクション処理システム）、特開平7-73143号公報（時間帯別優先順位制御のトランザクション処理システム）、特開平10-40117号公報（レスポンスを維持するタスク制御式のオンライントランザクション処理システム）等が知られている。

【0036】

【発明が解決しようとする課題】

複数のサービスプロバイダ（または顧客）のシステムをアウトソーシング形態で受託し、計算機資源を集中管理することにより全体としての処理効率を向上させるデータセンタなどの業態が盛んになりつつある。

【0037】

データセンタでは、委託元であるサービスプロバイダとの保証契約（サービスレベルアグリーメント：SLA）により、利用した各計算機資源（トランザクション量やそれに伴うCPU時間、データ量など）やサービスの保証形態に応じて課金を行い、対価を請求する。課金を廉価に押さえるためには、より少ない計算機資源（投資）でより多くのトランザクションを実行する必要がある。

【0038】

これに対し、上記従来のトランザクションモニタやメッセージブローカ等のトランザクション処理システムにおいては、単一のサービスプロバイダがクライア

ントにサービスを提供することを目的に作成されてきたため、一つのトランザクション処理システムを複数のサービスプロバイダが共用するような形態において、サービスプロバイダ間でのトランザクション資源（計算機資源）やスループットを調整するための配慮がなされていなかった。

【 0 0 3 9 】

このため、複数のクライアントからトランザクション処理要求があった場合には、計算機資源の適切な利用を図ることができず、スループットを保証するのが難しいという問題があった。

【 0 0 4 0 】

また、上記従来のメッセージブローカやトランザクションモニタでは、業務の更新や追加によるサービスフローの更新に際し、予備系プロセスを設けるか、あるいはルーティングを閉止するなどの処置が必要であり、計算機資源を有効に利用するための配慮が為されておらず、柔軟に運用することが難しいという問題があった。

【 0 0 4 1 】

そこで、本発明の目的は、上記 S L A を勘案したトランザクションの優先度制御、および計算機資源の割り当て制御を可能とすることにより、複数のサービスプロバイダに対するサービスを提供するのに好適なトランザクション処理システムを実現することにある。

【 0 0 4 2 】

【課題を解決するための手段】

本発明のトランザクション処理システムは、サービスに対する優先度制御を実現するために、

トランザクション処理システムが提供するサービスに対応して定義された優先度情報を格納する優先度情報保持手段と、提供するサービスに対してクライアントから送付された処理要求を、該提供するサービス毎に順序付けて格納するキューイング手段と、該キューイング手段から該格納した処理要求の滞留情報を取得する滞留情報取得手段と、該クライアントから該トランザクション処理システムに投入された処理要求の優先度を、優先度情報および処理要求の滞留情報を参照

して、その実行順序を決定する実行優先度決定手段とを有する。

【 0 0 4 3 】

より好ましくは、前記キューイング手段は、サービスを提供する顧客または利用者毎に処理要求を格納するキューを複数備え、また、優先度情報保持手段は、処理（実行するサービス）の種類と顧客または利用者に応じて定義付けられた優先度情報を有するものである。

【 0 0 4 4 】

また、本発明のトランザクション処理システムは、上記サービスに対する計算機資源の割り当て制御を行うために、各サービスに対するトランザクションスループットを検出する手段と、該各サービスに対してトランザクション処理プロセスを割り当てる手段と、を有し、該プロセスを割り当てる手段は、前記各サービスに対して、該取得した処理要求滞留情報と該検出したトランザクションスループットとを参照してプロセスの割り当てを決定する。

【 0 0 4 5 】

また、本発明のトランザクション処理システムは、各サービスを構成する実行モジュールの更新に応じて記憶手段への配置（ローディング）を行うために、

各サービスを構成する 1 つまたは複数の実行モジュールの識別子を格納する手段と、該各実行モジュールの更新を前記識別子に基づいて管理する更新管理手段と、を有し、該更新管理手段により該実行モジュールの更新が実行された場合には、該各サービスに対応するトランザクションの開始に先立って、該更新された実行モジュールを記憶手段へ配置する。

【 0 0 4 6 】

【発明の効果】

以上のように、本発明のトランザクション処理システムまたはメッセージブローカーでは、各サービスに対する優先度制御を実現するために、

トランザクション処理システムが提供するサービスに対応して定義された優先度情報と、クライアントから提供するサービスに対して送付された処理要求を該提供するサービス毎に順序付けて格納するキューイング手段から取得する処理要求の滞留情報とを勘案する。

【 0 0 4 7 】

このため、トランザクション処理システムが提供する顧客毎の各サービスについて、契約によって定義された条件を満たすようなトランザクションのスケジューリングが可能となり、より少ない計算機資源で、より多くのオンライントランザクションを、顧客との保証契約に応じたスループットを確保しながらリアルタイムで処理することが可能となり、複数の顧客の業務を集中して処理するデータセンタの信頼性と処理能力を向上させることができるのである。

【 0 0 4 8 】

また、本発明のトランザクション処理システムは、各サービスに対応するトランザクションのスループットを検出する手段と、該各サービスに対してトランザクション処理プロセスを割り当てる手段と、を有し、該プロセスを割り当てる手段は、各サービスに対して、取得した処理要求滞留情報と該検出したトランザクションスループットとを参照してプロセスの割り当ての決定を行う。

【 0 0 4 9 】

このため、トランザクション処理システムが提供する各サービスについて、契約によって定義された条件を満たすようなプロセスの割り当てが可能となって、顧客との保証契約に応じたスループットを確保しながらリアルタイムで処理することができ、複数の顧客の業務を集中して処理するデータセンタの信頼性と処理能力を向上させることができるのである。

【 0 0 5 0 】

さらに、本発明のトランザクション処理システムは、各サービスを構成する 1 つまたは複数の実行モジュールの識別子を格納する手段と、該各実行モジュールの更新を前記識別子に基づいて管理する更新管理手段と、を有し、該更新管理手段により該実行モジュールの更新が実行された場合には、該各サービスに対応するトランザクションの開始に先立って、該更新された実行モジュールを記憶手段へ配置する。

【 0 0 5 1 】

このため、実行モジュールが更新されているときには、各クライアントのトランザクション処理開始に先立って、更新された実行モジュールを記憶手段にロー

ドしてから実行するので、稼働状態を維持したまま業務の更新や追加を行うことが可能となって、柔軟性、可用性を高めることができ、また、実行モジュールの更新に際して、予備系プロセスを設けたりルーティングを閉止するなどの処置が不要となるため、計算機資源の有効利用を実現できる。

【 0 0 5 2 】

【発明の実施の形態】

以下、本発明の一実施形態を、添付図面に基づいて説明する。

【 0 0 5 3 】

1. ハードウェア構成

本発明の実施に好適な計算機システムのハードウェア構成例を図2に示す。システムは、1台または複数のクライアント端末701、702、1台または複数のトランザクション処理サーバ703、1台または複数のサービスプログラム実行用サーバ704、705より構成される。ただし、トランザクション処理サーバとサービスプログラム実行用サーバは同一の計算機であっても良い。

【 0 0 5 4 】

クライアント端末701、702はATM (Automatic Teller Machine) 端末やMicrosoft Windowsオペレーティングシステム、Linuxオペレーティングシステムなどの稼動するパーソナルコンピュータでも良い。

【 0 0 5 5 】

トランザクション処理サーバ703、サービスプログラム実行用サーバ704、705は、例えば、日立3500シリーズのようなUNIXサーバ、日立フローラ（登録商標）シリーズのようなWindows NTサーバ（登録商標）、あるいは日立MPシリーズのような大型汎用計算機で良い。クライアント、各サーバを結合する通信路710は、例えばEthernetのような汎用的なネットワークで良い。なお、トランザクション処理サーバ703、サービスプログラム実行用サーバ704、705には、図示しないメモリやハードディスク装置などの記憶手段が設置されている。

【 0 0 5 6 】

2. 発明の概要

本発明に好適な実施形態の詳細を説明するに先立ち、本発明全体の概要を図1を参照しながら説明する。

【0057】

クライアントプログラム101、102は、クライアント端末701、702上で稼動し、システムの末端利用者にインタフェースを提供するプログラムである。なお、クライアントは、トランザクション処理サーバ703と接続する顧客毎のクライアント端末701、702からなる。

【0058】

ATMの制御プログラムや、パーソナルコンピュータ用のクライアントプログラムがこれに相当する。クライアントプログラムはwebブラウザであっても良い。クライアントプログラムは末端利用者の入力を受け付けて、メッセージを組み立て、トランザクションモニタ120に送信する。

【0059】

トランザクションモニタ120は本発明の中心を構成する部分である。前記従来のトランザクションモニタとは異なり、本発明のトランザクションモニタは、複数のサービスプロバイダ（顧客、以下同様）に対するメッセージ（処理要求）を受け付ける。図1では、サービスプロバイダの数を2と仮定している。

【0060】

SLAデータベース（優先度情報データベース）113は、各サービスプロバイダとのサービスレベル（優先度情報、許容待ち時間）に関する契約情報（SLA）を格納する。例えば、「サービスプロバイダAのトランザクションは、10秒以内に処理されること」などの契約内容に基づいて、許容待ち時間が10msec、優先度がU、Lなどの情報が本データベースに格納される。

【0061】

メッセージディクショナリ114には、サービスプロバイダ毎のメッセージのフォーマットを定義する。例えば「プロバイダAのメッセージの第10バイトから第20バイトまでの間は、顧客口座番号である」などの定義が格納される。

【0062】

サービスフロー定義115には、サービスプロバイダ毎のサービスフローの定義が格納される。実行モジュールライブラリ116には、サービスフローの各サービスノードに対応した実行モジュール群が格納される。

【0063】

プリプロセッサ103は、クライアントプログラム101、102から受け付けたメッセージを解釈し、該メッセージがどのサービスプロバイダに属するかを判定する。

【0064】

リクエストキュー110、111は、該トランザクションモニタ120を利用するサービスプロバイダ毎（顧客毎）に設けられ、それぞれのサービスプロバイダに対して送られたリクエストを格納する。図1ではサービスプロバイダの数を2と仮定しているので、リクエストキューは110、111の2本が存在する。

【0065】

キューイング状況検出部112は、リクエストキュー110、111を監視し、その状況（滞留リクエスト数やスループット）を取得する。

【0066】

スケジューラ104は、キューイング状況検出部112から得られるキューイング状況と、SLAデータベース113に格納されたSLA契約とを勘案してスケジューリングの優先度を決定する。スケジューラ104はまた、サービスプロバイダ毎に割り当てるプロセス108、109の数を管理し、SLA契約を満たすために適切な数のプロセス数を決定する。

【0067】

スケジューラ104により取り上げられたメッセージは、ダイナミックローダ105に送られる。

【0068】

ダイナミックローダ105は、サービスフロー定義115を参照し、現在のメッセージに対応するサービスフローを決定する。

【0069】

実行モジュール管理部106は、実行モジュールライブラリ116を監視し、

変更が加えられた場合にこれを検出する。ダイナミックローダ 1 0 5 はこの検出結果を参照し、現在のメッセージに対応するサービスの実行に必要なサービスノードが当該プロセス中にすでにローディングされているかを判別し、ローディングされていない（あるいは古いモジュールがローディングされている場合には、新たなモジュールをローディングする。サービスフロー実行部 1 0 7 は、スケジューリングされたサービスフローを実行する。

【 0 0 7 0 】

以下に、本発明を構成する各要素の詳細な実施形態について説明する。

【 0 0 7 1 】

3. S L A データベース

図 3 を参照して、S L A データベース 1 1 3 の構成例を説明する。

【 0 0 7 2 】

S L A データベース 1 1 3 は表の形でディスク上に格納され、トランザクションモニタ 1 2 0 を運用するデータセンタが、顧客（サービスプロバイダ）から受託した契約内容を蓄積する。

【 0 0 7 3 】

表の列はサービスプロバイダ名称 8 0 1、サービス名称（処理の種類） 8 0 2、クラス 8 0 3、上限値 8 0 4、優先度 8 0 5 を含む。

【 0 0 7 4 】

サービスプロバイダ名称 8 0 1 は、本トランザクションモニタの処理契約対象となっているサービスプロバイダを特定する名称である。本カラムは、一意の名称であれば任意の文字列で良い。

【 0 0 7 5 】

サービス名称 8 0 2 は、当該サービスプロバイダが本トランザクションモニタで提供するサービスの名称である。クラス 8 0 3 は、サービスプロバイダとの契約の種別を表す。「B. E.」はベストエフォートの略であり、リソースに空きがある場合にはトランザクションをスケジュールする契約を示す。リソースが込み合っていた場合、トランザクションは長い間待たされる可能性がある。「U. L.」はアッパーリミットの略であり、トランザクション待ち合わせの上限時間

を決める契約を示す。

【0076】

上限値804は、U. L. 契約における上限時間を示す。B. E. 契約時にはこのカラムは意味を持たない。

【0077】

優先度805は、U. L. 契約されたサービスの間での優先順位を示す。リソースが込み合ってすべてのU. L. 契約を満たせなくなった場合に、優先度の高い順にスケジューリングを行う。なお、優先度805は、サービスプロバイダとの契約に応じて定めても良いし、データセンタ側が顧客であるサービスプロバイダやサービスの優先度を判断して独自に定めても良い。

【0078】

上記図3はSLAの基本的なデータベースの構成であるが、これに加えてデータセンタ独自の項目を設定することも可能であり、例えば、提供するサービスの処理負荷などに応じた優先度などを設定できる。

【0079】

したがって、SLAデータベース113（優先度情報保持手段）においては、提供するサービス（顧客、処理＝サービスフローの種類）に対応して、優先度、上限値（許容待ち時間）が定義されており、この定義は、図示しない入力手段を介してオペレータなどが設定、格納するもので、プリプロセッサ103及びスケジューラ104が、SLAデータベース113の優先度情報を参照し、スケジューラ104（実行優先度決定手段）は後述のように、サービスを識別する情報をキーとして、サービスプロバイダ名、サービス名を検索し、SLAデータベース113の優先度情報を読み込む。

【0080】

4. メッセージディクショナリ

図4および図6を参照して、メッセージディクショナリ114の構成例を説明する。メッセージディクショナリ114は、メッセージ形式の、サービスプロバイダおよびサービス毎の定義を格納する。

【0081】

トランザクションモニタ 1 2 0 の受け付けるメッセージは、固定部（図 6 の 1 0 0 1、1 0 0 2）と可変部（図 6 の 1 0 0 3）より構成される。固定部はトランザクションモニタ 1 2 0 に固有のメッセージフィールドであり、一方、可変部は、サービスプロバイダ、およびサービス毎に異なるメッセージのフィールドである。

【 0 0 8 2 】

図 6 のメッセージ構成に対応して、メッセージディクショナリ 1 1 4 も固定部定義 { 図 4 (a) } と可変部定義 { 図 4 (b)、(c)、(d) } を有する。

【 0 0 8 3 】

図 4 (a) の例では、固定部定義は開始バイト (9 0 1)、長さ (9 0 2)、タイプ (9 0 3) のカラムを有する。メッセージの第 0 バイトから長さ 3 2 バイト分のフィールドにサービスプロバイダ名称が、第 3 2 バイトから長さ 3 2 バイト分のフィールドにサービス名称が格納されることを示す。第 6 4 バイト以降のフィールドは、可変部に相当する。

【 0 0 8 4 】

可変部定義は、可変部インデクス定義 { 図 4 (b) } と、可変部フィールド定義 { 図 4 (c)、(d) } を組み合わせて行う。

【 0 0 8 5 】

可変部インデクス定義は、メッセージ固定部 1 0 0 1、1 0 0 2 に入れられたサービスプロバイダ名 9 0 5 およびサービス名 9 0 6（サービスを識別する情報）をキーとして、可変部フィールド定義のインデクスを検索するためのテーブルである。図 4 (b) では、例えば「サービスプロバイダ A」の「サービス A 1」のインデクスは「1」である。

【 0 0 8 6 】

これと同じテーブルインデクス (= 「1」) をもつ可変部フィールド定義 { 図 4 (c) } が、「サービス A 1」に関する定義となる。同様に、「サービスプロバイダ A」の「サービス A 2」のインデクスは「2」である。これと同じテーブルインデクスをもつ可変部フィールド定義 { 図 4 (d) } が、「サービス A 2」に関する定義となる。

【0087】

各テーブルインデクスは、開始バイト、長さ、データタイプの列に応じたフィールドが設定されており、図4（c）では、メッセージの第64バイトから4バイト分のフィールドに口座番号が、第68バイトから12バイトのフィールドにタイムスタンプが、第80バイトから8バイト分のフィールドに引き出し額が格納されることを示す。図4（d）も同様であるが、第80バイトから8バイト分のフィールドは現在残高に対応する。

【0088】

以上のような定義により、あるメッセージがトランザクションモニタ120に入力された場合に、固定部1001、1002によってそのメッセージがどのサービスプロバイダのどのサービスに属するかが判別できる。また、可変部1003によって該サービスに対するパラメータを指定することができる。

【0089】

5. サービスフロー定義及びサービスフロー実行部

サービスフロー実行部107は、図5に示すように、メッセージを用いて個別の処理を連結することにより形成する。単一の目的を持った処理（処理ノード）を複数組み合わせることによって複雑な機能を実現することができる。

【0090】

図5（a）では、サービスフローが601～605の5個の処理ノードからなる場合を示し、図中の矢印はノード間のメッセージの流れを示す。

【0091】

ノード601はトランザクションモニタを経由して端末からのメッセージを受け取り、これを処理ノード602に渡す。処理ノード602は、メッセージを参照しつつ利用者によって定義された処理を行い、必要に応じてメッセージを変形してさらに下流のノード603、604、605にメッセージを渡す。

【0092】

603はメッセージの変換ノードであり、出力先のサービスプログラムの形式に合わせて、メッセージ中のコード変換（例えばEBCDICコードからASCIIコードへの変換）を行う。604、605は出力ノードであり、トランザク

ションモニタ経由で外部のサービスプログラムに対してメッセージを送出する。

【0093】

図5（b）には、サービスフローの定義情報115を示す。

【0094】

620～624は、それぞれ601～605のノードに対する定義情報である。サービス名は、各ノードが属するサービスの名称を指定する。ノード名称には任意のノード名称を、フロー内で一意になるように指定する。ノード種類は、入力ノード、処理ノード、変換ノード、出力ノードなど、当該メッセージブローカシステムにて提供されるノードの種類から適当なものを選んで指定する。入力先、出力先には、当該ノードに対するメッセージの入出力先ノード名称を指定する。例えばノードB602はノードA601からメッセージを受け入れ、ノードC603およびノードE605にメッセージを出力する。処理ノード、変換ノードには、個別に処理内容を指定する。

【0095】

処理内容の指定は、定義情報620～624の「モジュール」カラムに、対応する処理モジュールの名称を格納することで可能となる。入力・出力ノードのような定型処理を行うノードに対しては、予め固定されたモジュールを用いるため、名称の指定は必要ない。

【0096】

このサービスフロー定義115とサービスフロー実行部107により、トランザクションモニタ上でサービスフローの実行を可能とすることにより、複数のサービスプログラムを組み合わせより複雑なサービスを提供するメッセージブローカを構築することができる。

【0097】

6. 実行モジュールライブラリ

実行モジュールライブラリ116は、サービスフロー中の各サービスノードを実行するのに必要な実行モジュール群を格納する。各実行モジュールは例えばUNIXのファイル形式でディスクに格納することができる。ファイル名称をサービスフロー定義中に現れるモジュール名称と一致させることにより、サービスフ

ローから実行モジュールの検索が可能となる。

【0098】

また、実行モジュールは実行時に動的にローディングが可能な形式、例えばUNIXのDLL (Dynamic Loading Library) 形式で作成する。

【0099】

7. リクエストキュー

リクエストキュー110、111は、トランザクションモニタ120に入力されたメッセージを入力順に格納するデータ構造である。

【0100】

リクエストキュー110、111は、トランザクションモニタ120に登録されたサービスプロバイダのサービス毎に一つずつ作成する。リクエストキューの構造の例を図7に示す。

【0101】

図7において、リクエストキューは、キュー毎に一つ存在するリクエストヘッダ1114～1116と、そこからリスト構造でつながる複数のリクエスト構造体1101～1104よりなる。

【0102】

リクエストヘッダは、サービス情報1114、SLA情報1115、後方チェイン1116、キューの先頭ポインタ1117およびキュー情報1118の各フィールドを有する。

【0103】

サービス情報1114はキューに割り当てられたサービスプロバイダおよびサービス名称を格納する。SLA情報1115はSLAデータベース113に格納されたSLA定義を、サービスプロバイダ名称およびサービス名称をキーとして検索することにより取得し、リクエストヘッダに格納する。

【0104】

後方チェイン1116は、キューが複数存在する場合に、リクエストヘッダをリスト構造にて連結するためのポインタを格納する。図中、1130～1132

は、複数のキューが後方ポインタを用いて連結された状態を示す。

【0105】

キューの先頭ポインタ1117は、キューの先頭リクエスト構造体（最初に作成された構造体）へのポインタを格納する。キュー情報1118は、キューにつながれたリクエストの状況を格納するフィールドであり、その使用法は後述する。

【0106】

各リクエスト構造体には、4つのフィールド1110～1113が存在する。タイムスタンプ1110は、リクエストの生成時刻を示すフィールドである。前方チェーン1113および後方チェーン1114は、リクエスト構造体を相互に連結してキューを形成するためのポインタを格納する。メッセージポインタ1115は、メッセージ本体の格納領域に対するポインタを格納する。

【0107】

1101～1104は、前方チェーン、後方チェーンを用いてリクエスト構造体がキューを構成した状態の例を示す。1120～1123は、各リクエストに対応するメッセージの格納領域であり、対応するリクエスト構造体よりポイントされる。

【0108】

8. プリプロセッサ

プリプロセッサ103は、トランザクションモニタ120に入力されたメッセージをメッセージディクショナリ114の内容と突き合わせ、当該メッセージがどのサービスプロバイダのどのサービスに対応するかを解析して、適切なリクエストキュー110、111に格納する。

【0109】

図8にプリプロセッサの動作フローの例を示す。

【0110】

プリプロセッサ103は起動時に、メッセージディクショナリ114からメッセージ固定部（図6中1001、1002）の情報を読み込む（1201）。

【0111】

次に、トランザクションモニタ 1 2 0 のサービス終了まで、クライアントからのメッセージを受け付けるループ 1 2 0 2 に入り、メッセージ入力待ち状態となる (1 2 0 3) 。サービス終了は、特殊なメッセージを受け付けることによってループ 1 2 0 2 から抜けても良いし、割り込み処理によってループを終了しても良い。

【 0 1 1 2 】

メッセージの入力待ちは、例えば UNIX の `accept` システムコールなどにより実現できる。メッセージが入力されると、すでに 1 2 0 1 にて読み込んだメッセージ固定部情報を用いて、メッセージに対応するサービスプロバイダ名、サービス名を切り出す (1 2 0 4) 。

【 0 1 1 3 】

次に、リクエストヘッダを順次サーチして (1 2 0 5) 、取得したサービスプロバイダ名、サービス名に該当するキューを検索し (1 2 0 6) 、そのキューに入力されたメッセージ (電文) を登録する。メッセージの登録は、メッセージとサービスプロバイダ名、サービス名を含むメッセージ構造体 (図 7 中 1 1 1 0 ~ 1 1 1 3) を新たに作成し、キュー構造 (図 7 中 1 1 0 1 から 1 1 0 4) の最後尾に、作成した構造体をポインタ操作により挿入することにより実現できる。

【 0 1 1 4 】

9. キューイング状況検出部

キューイング状況検出部 1 1 2 は、各リクエストキュー 1 1 0、1 1 1 の状況を監視し、スケジューラ 1 0 4 がスケジュールすべきリクエストを選択するとともに、適当な量のリソースを各サービスに配分するために必要な情報を抽出する。

【 0 1 1 5 】

キューイング状況検出部 1 1 2 は、トランザクションモニタ 1 2 0 またはサーバのオペレーティングシステムによって一定時間毎に起動され、所定の処理を行う。一定時間毎の起動には、例えば UNIX オペレーティングシステムの `signalarm` システムコールを用いることができる。

【 0 1 1 6 】

毎回の起動によって実行される処理フローの例を図9に示す。

【0117】

各リクエストヘッダについて（1301）、そこからポイントされるリクエスト構造体をスキャンし（1302）、キュー中のリクエスト数をカウントアップする（1303）と同時に、当該キューのリクエスト構造体に格納されたタイムスタンプのうち最も古いものを求める。

【0118】

このようにして取得したリクエスト数と、最も古いタイムスタンプとを、リクエストヘッダのキュー情報フィールド（図7の1118）に格納する。

【0119】

10. スケジューラ

スケジューラ104は、キューイング状況検出部112によって抽出された情報に基づき、リクエストのスケジューリングを実行する。

【0120】

スケジューリングは、SLAのクラスがU. L.（アッパーリミット契約）であるリクエストを優先し、余裕がある場合にB. E. クラス（ベストエフォート契約）のリクエストをスケジューリングする。各々のクラスのうちでは、タイムスタンプがもっとも古いリクエストから優先的に処理を行う。

【0121】

図10に、スケジューリングすべきリクエストの選択処理フローの具体例を示す。

【0122】

まず、各一時変数を初期化する（1401）。ここで、TulはU. L. クラスサービスプロバイダに属するリクエストのタイムスタンプを格納する一時変数、Tbeは同様にB. E. クラスのリクエストのタイムスタンプを格納する一時変数である。Pul、PbeはそれぞれU. L. クラス、B. E. クラスのリクエストに対するポインタを格納する一時変数である。

【0123】

次に、リクエストヘッダリスト（図7中1130～1132）に格納された各

ヘッダについて(1402)、ヘッダ中のSLA情報(図7中1115)を参照して、当該ヘッダがU. L. クラスかB. E. クラスかを判別する(1403)

。

【0124】

U. L. クラスであった場合には、T u lに格納されたこれまでのタイムスタンプの最小値と、当該リクエストヘッダに格納されたキュー情報(図7中1118)より得られるキューの最古タイムスタンプとを比較する(1404)。当該タイムスタンプの方が古かった(小さかった)場合には、一時変数T u lを入れ替えるとともに(1405)、当該リクエストヘッダへのポインタを一時変数P u lに格納する。

【0125】

一方、上記判別1403において当該ヘッダがB. E. クラスに属すると判定された場合には、一時変数T b eとP b eとを用いて同様の処理を行う(1407~1409)。以上の処理によって、U. L. クラス、B. E. クラスのそれぞれについて最古のタイムスタンプと、それに対応するリクエストヘッダを求めることができる。

【0126】

次いで、U. L. クラス、B. E. クラスのいずれのリクエストをスケジュールすべきかを決定する。

【0127】

まず、P u lかP b eのいずれかがNULLであった場合には、NULLでない方のリクエストをスケジューリングする(1410~1412)。

【0128】

双方ともNULLでなかった場合には、以下の式1)を評価する(1413)

。

$$T u l < ((\text{現在時刻} - \text{上限値}) + e) \quad \dots (1)$$

ここに、現在時刻は本処理を実行中の時刻である。上限値はSLAデータベース113に定義されたSLA契約における当該サービスの上限時間(図3中804)であり、リクエストヘッダ(図7中1115)中のSAL情報を参照して得ら

れる。また、 e は本トランザクションモニタ 120 の運用者により決定されるオフセットである。

【0129】

上記 (1) 式は、U. L. クラスの最も古いタイムスタンプを持つリクエストが、SLA 契約における処理の後れの上限值からある時間幅 (e) 以内に存在するか否かを調べることに相当する。存在する場合には、U. L. クラスを優先し、このリクエストをスケジューリング対象とする (1414)。一方、時間幅 (e) 以内に存在しない場合には、U. L. クラスの処理に余裕があるので、B. E. クラスのリクエストのうちもっとも古いタイムスタンプをもつものをスケジューリングする (1415)。

【0130】

11. ダイナミックローダ

ダイナミックローダ 105 は、スケジューラ 104 によるリクエストのスケジューリング結果を受けて、プロセスの起動と実行モジュールのローディングを行う。

【0131】

ダイナミックローダ 105 内部には、サービスフロー実行部 107 にて起動されたプロセスの状態を管理するためのプロセス管理情報を有する。

【0132】

図 11 を用いて、プロセス管理情報の構成例を説明する。

【0133】

プロセス管理情報は、各サービスに対してどのプロセスが対応しており、その各プロセスに対してどの実行モジュールがローディングされているかを管理する。

【0134】

サービス構造体は 1501 ~ 1503 のフィールドを有し、サービス名称を格納する。この構造体を 1521、1522 のように連結することにより、サービス対応のリスト構造 (サービスチェイン) を作成する。

【0135】

各サービス構造体1521、1522からは、プロセス構造体1531、1532がプロセスポインタ1502によってポイントされる。

【0136】

プロセス構造体1531、1532は、4つのフィールドを有し、プロセスID1504、実行モジュールへのポインタ1503、該プロセスが使用中か否かを示すフラグ1504および後方ポインタ1505を格納する。

【0137】

本構造体を1531、1532のように連結してリスト構造（プロセスチェーン）を形成する。さらに、プロセス構造体1531、1532からは実行モジュール構造体1541～1543及び1551～1553がポイントされる。

【0138】

実行モジュール構造体は、モジュール名称1508、後方ポインタ1509および実行モジュールのバージョンを示すカウンタ情報1510を格納する。実行モジュール構造体を1541～1543（あるいは1551～1553）のように連結してリスト構造（モジュールチェーン）を形成する。

【0139】

次に、図12を用いてダイナミックローダの処理フローを説明する。

【0140】

まず、プロセス管理情報中のサービスチェーン（図11中1521、1522）をたどり、スケジューリング対象サービスがチェーン中に存在するかを調べる（1602）。

【0141】

存在する場合には、そのサービスを処理するための少なくとも一つのプロセスがすでに起動されているので、プロセスチェーン（図11中1531、1532）をたどって、そのうち未使用のものをサーチする（1603、1604）。

【0142】

未使用のプロセスが存在する場合には、実行モジュール管理部106中の実行モジュールテーブル1700を共有ロックし、そのプロセスを構成するモジュールチェーン（図11中1541～1543）をたどり、各実行モジュールが前回

のローディング以降変更または更新されたかを調べる（1607、1608）。

【0143】

実行モジュール管理部106、実行モジュール管理テーブル1700の詳細については後述する。変更が検出された場合には、該モジュールをローディングする（1609）。

【0144】

スケジューリング対象のサービスがチェイン中に存在しない場合（1605）、またはプロセスチェイン中に未使用のプロセスが無い場合（1606）には、新たにプロセスを起動して、必要な実行モジュールのローディングを行う（1621）。

【0145】

まずプロセス起動を行って、そのIDをプロセスチェインに登録し（1622）、サービスフロー定義115中のサービスフロー定義テーブルの各カラムについて（1623）、各モジュールが現在着目するサービスに属するか否かを判定し（1624）、属する場合にはローディングを行う（1625）。なお、プロセスのIDとしては、たとえばUNIXオペレーティングシステムで付されるpidを利用すれば良い。

【0146】

12. 実行モジュール管理部

実行モジュール管理部106は、実行モジュールライブラリ116中の実行モジュールの追加、更新、削除を管理する。実行モジュールの状態を保持するデータ構造として、実行モジュール状態テーブルを有する。

【0147】

実行モジュール状態テーブルの例を図13に示す。

【0148】

状態テーブル1700のカラム1701～1705は実行モジュールライブラリ116に格納された各実行モジュールに対応する。各々の実行モジュールに対し、実行モジュール名称および更新カウンタ情報（識別子）を格納する。

【0149】

更新カウンタは当該実行モジュールの更新数を表す整数であり、モジュール新規登録時には1を格納し、モジュールが更新される毎に1ずつインクリメントする。

【0150】

さらに、実行モジュール状態テーブル1700にはロックフィールド1710が付随する。本フィールドはテーブルのロック状況を格納し、N（ロック無し）、S（共有ロック中）、E（排他ロック中）の3通りの値をとる。

【0151】

ここで、ダイナミックローダ105が本状態テーブルを用いて実行モジュールの変更状態を検出するステップ（図12中1608）について、図14を用いて詳細を説明する。

【0152】

まず、ダイナミックローダ105は、実行モジュール状態テーブル1700のロックフィールドを共有ロックし（図12の1606）、ループ1607にて現在着目している実行モジュールの名称を対応するモジュール構造体（例えば1541）から取得する（1801）。

【0153】

次に、この名称を用いて実行モジュール状態テーブルをルックアップし（1802）、対応する更新カウンタを取得する（1803）。さらに、取得したカウンタの値とモジュール構造体（例えば1541）中のバージョンカウンタ（図11の1510）の値とを比較する（1804）。

【0154】

更新カウンタとバージョンカウンタの値が同じであった場合には、前回ローディング時から実行モジュールに変更はなかったと判断し、実行モジュールの再ローディングは行わない。

【0155】

一方、更新カウンタがバージョンカウンタよりも大きかった場合には、前回ローディング時以降に実行モジュールが変更されたとみなし、実行モジュールを再ローディングするとともに、バージョンカウンタに更新カウンタの値を代入する

【0156】

一方、実行モジュールライブラリにモジュールを更新する場合の、実行モジュール管理部106の処理フローを図15を用いて説明する。

【0157】

まず、実行モジュール状態テーブル1700のロックフィールド1710を排他ロック（「S」）する。

【0158】

続いて、更新対象となる実行モジュールライブラリの名称を、トランザクションモニタ120より取得する。

【0159】

この名称は、例えば管理者がトランザクションモニタの管理コンソール（図2中720）から入力する情報から取得することができる。続いて、実行モジュール状態テーブル1700をサーチし、更新対象モジュール名称と同じ名称を持つカラムを見出し（1904）、そのカラムの更新カウンタをインクリメントする（1905）。最後に、実行モジュール状態テーブル1700の排他ロックを解除（「N」）する。

【0160】

1.3. 作用

以上のような構成により、リクエストキュー110、111を、トランザクションモニタ120に登録されたサービスプロバイダのサービス毎に一つずつ設けるのに加え、入力されたメッセージをメッセージディクショナリ114の内容に基づいて適切なリクエストキュー110、111に送るプリプロセッサ103と、リクエストキュー110、111の状況を監視して、スケジューラ104がスケジュールすべきリクエストを選択するキューイング状況検出部112と、SLAデータベース113に格納された複数のサービスプロバイダ（顧客）とのサービスの優先度を表す情報（サービスレベルに関する契約情報）をキーとして、スケジューラ104がリクエストを制御することにより、複数の顧客でひとつのトランザクションモニタ120（またはメッセージブローカ）を共有しながらも、

所定の優先度またはリソースの状態に応じて最適なリソースの割り当てが行われるため、適切なスループットを保証することができるのである。

【 0 1 6 1 】

したがって、複数のサービスプロバイダのシステムをアウトソーシング形態で受託して、計算機資源を集中管理するデータセンタなどに適用すれば、より少ない計算機資源（リソース）で、より多くのオンライントランザクションを、顧客との保証契約に応じたスループットを確保しながらリアルタイムで処理することが可能となり、複数の顧客の業務を集中して処理するデータセンタの信頼性と処理能力を向上させることができるのである。

【 0 1 6 2 】

また、複数のサービスプロバイダのサービス毎に、必要なプロセスを実行するダイナミックローダ 1 0 5 は、各プロセスを構成する実行モジュールが更新されたか否かを検出する実行モジュール管理部 1 0 6 に基づいて、更新があったモジュールを一括してローディングした後にトランザクション処理を行うようにしたため、実行モジュールのルーティング停止や予備のプロセスを設けることなく、トランザクションモニタ 1 2 0 稼働中にサービスの変更を随時行うことが可能となり、計算機資源の有効利用を維持しながらも顧客の業務ロジックの追加、変更を容易に行って、柔軟性、可用性に富んだトランザクションモニタ 1 2 0 またはメッセージブローカを構築することができ、システムの運用を簡易にすることができるのである。

【 0 1 6 3 】

図 1 6、図 1 7 は、第 2 の実施形態を示す。

【 0 1 6 4 】

前記第 1 の実施例においては、サービスフロー実行部 1 0 7 の有するプロセス数に十分な余裕があり、スケジューラは必ずリクエストのスケジューリングが可能であることを想定した。

【 0 1 6 5 】

これに対し、本実施形態では、計算機資源の制約からプロセス数が必ずしも十分に確保できず、サービス間でプロセスのトレードオフが必要となる場合を想定

し、より一般的な場合を想定する。

【0166】

本実施形態では、スケジューラ104に代えて、プロセス管理部2001を設ける。なお、その他は前記第1実施形態と同様である。

【0167】

プロセス管理部2001の役割は、各サービスのリクエストキュー110、111の状態とSLA契約とから、その処理に必要とされるプロセス数を推定し、ダイナミックローダ105を制御することにある。

【0168】

各プロセスは、現在処理中のトランザクションが修了するとリクエスト受け付け状態に入り、対応するリクエストキュー110、111からリクエストを取り出して、次のトランザクションの処理を行う。

【0169】

プロセス管理部2001における処理の流れを、図17に基づいて説明する。

【0170】

プロセス管理部2001では、システム開始時にSLAデータベース113から各サービスに関する上記図3のSLA情報を取得する。プロセス管理部2001は、システム稼働中は定期的にキューとプロセスの状況を監視し(2103)、以下の処理を行う。

【0171】

まず、キューイング状況検出部112から各サービスについてのキュー情報を取得する(2102)。キュー情報は、滞留リクエストの個数および最古のタイムスタンプであり、上記図9にて説明した通り、キューイング状況検出部112で抽出されたリクエストヘッダのキュー情報フィールド1118を参照して取得できる。

【0172】

次いで、各サービスについて(2104)、該サービスに対応するトランザクションの開始時刻、終了時刻、プロセス数をサービスフロー実行部から取得し、該サービスに対応するトランザクションのスループットを計算する。通常一つの

サービスに対応するプロセスは複数あるので（図 1 6 中 1 0 8、1 0 9）、当該サービスに対するトータルスループットは、各プロセスが処理するトランザクションに要する時間の逆数の総和をとることにより求められる。

【0 1 7 3】

一方、取得したキュー情報から、前回と今回のキュー長（キュー中の滞留リクエスト数）の差分を求め、リクエスト到着頻度を計算する。リクエスト到着頻度は、キュー長の差分を、時間間隔で除することにより計算できる。

【0 1 7 4】

あるいは、各トランザクションの開始時刻と終了時刻の差を取得し、その逆数に、各サービスに割り当てたプロセス数を乗ずることでスループットを求めてもよい。

【0 1 7 5】

このようにして求めたトータルスループットとリクエスト到着頻度を比較することにより、該サービスに対するスループットの充足度を推定することができる。

【0 1 7 6】

すなわち、トータルスループットがリクエスト到着頻度より大きければキューの長さは時間とともに減少し、小さければ増大する。ここではスループット充足度を、トータルスループットをリクエスト到着頻度で除することにより求める（2 1 0 9）。

【0 1 7 7】

以上で各サービスに対するスループットの充足度が求まったので、次に、各サービスに対するプロセス数を変更し、スループットが各サービスに対して最適に配分されるように制御する。SLA 契約で優先度の高いサービス順に（2 1 0 8）、スループット充足度を 1 以上にするために必要な新プロセス数を計算する。新プロセス数が現プロセス数より大きい場合には、新たに差分のプロセスを起動し、ダイナミックローダ 1 0 5 を通じて必要な実行モジュールのローディングを行ってそれらを受信待ち状態におく（2 1 1 2）。

【0 1 7 8】

トランザクションモニタ全体でプロセス数に制限があり、必要なプロセスをすべて起動できない場合には、可能な範囲で最大のプロセス数を起動する（2113）。一方、スループット充足度に余裕がある場合には、余裕分のプロセスを停止してシステムリソースを解放する（2114）。

【0179】

このようなスケジューリングによれば、SLA契約で優先度の高いサービスに優先的にプロセスが配分され、契約が満たされる確立を高めると同時に、スループット充足度に余裕のある場合には優先度の低いサービスに資源を回すことが可能となる。

【0180】

したがって、計算機資源の制約からプロセス数が必ずしも十分に確保できない状態においても、SLA契約に応じたスループットを確保できるように、計算機リソースの制御を行うことができ、信頼性を向上させることができる。

【0181】

なお、リクエスト到着頻度に変動が大きい場合など、図17の処理ではプロセスの起動、停止が頻発する可能性がある。一度起動したプロセスは一定期間停止させないなど、履歴を考慮した制御を行うことにより、プロセス数の過度の変動を抑止することができる。

【0182】

また、図17の処理では、優先度の高いリクエストが多数投入された場合に、優先度の低いリクエストの実行が滞る可能性がある。このような場合には、各サービスに対する最小プロセス数を予め決めておき、これを下回らない範囲でプロセス数の増減を行えば良い。

【0183】

さらに、上記第2実施形態の特徴的な部分としては、1つまたは複数のサービスを提供し、該提供する各サービスには1つまたは複数のクライアントが接続可能なトランザクション処理であって、該クライアントから前記提供するサービスに対して送付された処理要求を、提供するサービス毎に順序付けて格納するキューイング手段（110、111）と、キューイング手段に格納した処理要求の滞

留情報を取得する滞留情報取得手段（キューイング状況検出部 1 1 2）と、前記各サービスに対するトランザクションのスループットを検出するスループット検出手段及び前記各サービスに対してトランザクションの処理プロセスを割り当てるプロセス割り当て手段（プロセス管理部 2 0 0 1）と、を有し、このプロセス割り当て手段は、前記各サービスに対して、取得した処理要求滞留情報と検出したトランザクションスループットとを参照してプロセスの割り当てを決定するプログラムという点である。

【 0 1 8 4 】

より詳しくは、プロセスの割り当てを、処理要求滞留情報から算出した単位時間における処理要求到着頻度と、トランザクションスループットとの大小を比較し、処理要求到着頻度がトランザクションスループットより大きい場合には割り当てるプロセス数を増加させ、一方、処理要求到着頻度がトランザクションスループットより小さい場合には割り当てるプロセス数を減少させるというプログラムという点である。

【図面の簡単な説明】

【図 1】

本発明実施例の全体図。

【図 2】

本発明のハードウェア構成図。

【図 3】

S L A データベースの説明図。

【図 4】

メッセージディクショナリの説明図。

【図 5】

サービスフローの説明図で、（a）はノードとサービスプログラムの関係を示し、（B）はサービス名に対応したノード名、ノード種類、入力先、出力先、モジュールの関係を示す。

【図 6】

メッセージの説明図。

【図 7】

リクエストキューの構成説明図。

【図 8】

リクエストキューの動作説明図。

【図 9】

キューイング状況検出部の動作説明図。

【図 1 0】

スケジューラの動作説明図。

【図 1 1】

プロセス管理情報の構成説明図。

【図 1 2】

ダイナミックローダの動作説明図。

【図 1 3】

実行モジュール状態テーブルの構成説明図。

【図 1 4】

ダイナミックローダの動作詳細説明図。

【図 1 5】

実行モジュール管理部の動作説明図。

【図 1 6】

第 2 の実施形態を示す構成図。

【図 1 7】

プロセス管理部の動作説明図。

【図 1 8】

従来のトランザクション処理システムの説明図。

【図 1 9】

従来のプロセス数制御の説明図で、（A）はリクエストがひとつの場合、（B）はリクエストが溜まった場合、（C）はリクエストが減少した場合を示す。

【図 2 0】

従来の優先度制御の説明図。

【図 2 1】

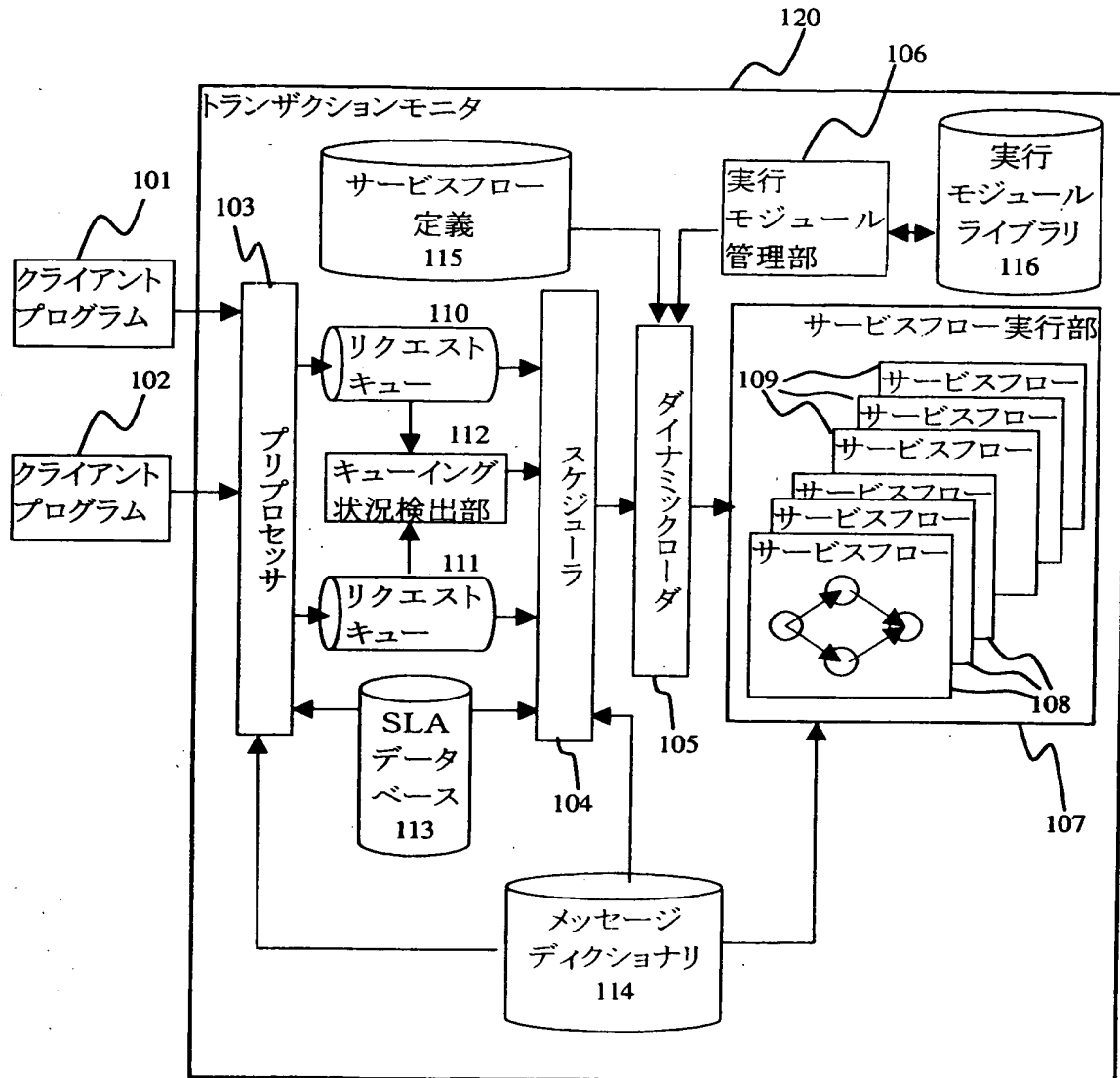
従来のメッセージブローカの説明図。

【符号の説明】

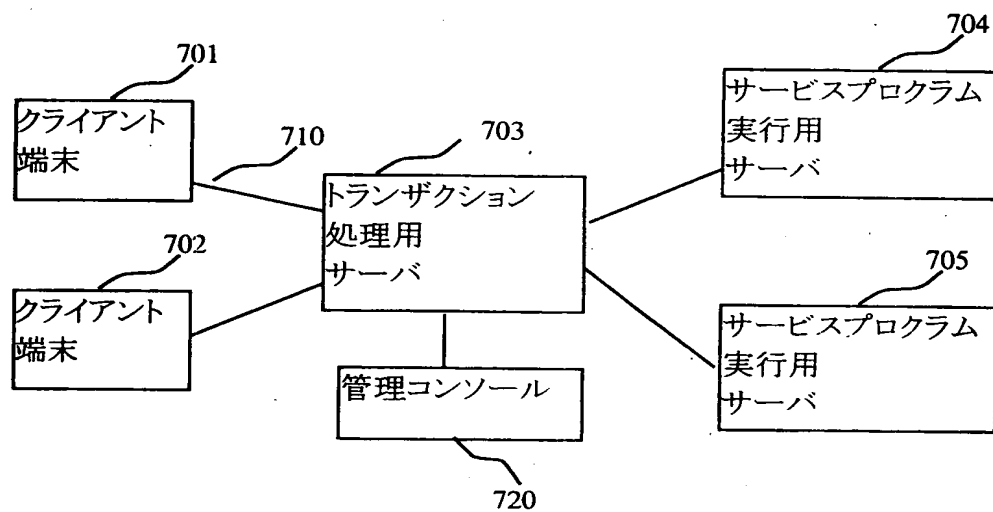
- 1 0 1、1 0 2 クライアントプログラム
- 1 0 3 プリプロセッサ
- 1 0 4 スケジューラ
- 1 0 5 ダイナミックローダ
- 1 0 6 実行モジュール管理部
- 1 0 7 サービスフロー管理部
- 1 0 8、1 0 9 サービスフロー
- 1 1 0、1 1 1 リクエストキュー
- 1 1 2 キューイング状況検出部
- 1 1 3 S L A データベース
- 1 1 4 メッセージディクショナリ
- 1 1 5 サービスフロー定義
- 1 1 6 実行モジュールライブラリ
- 1 2 0 トランザクションモニタ

【書類名】 図面

【図 1】



【図 2】



【図 3】

サービスプロバイダ名称	サービス名称	クラス	上限値	優先度
プロバイダ A	サービス A1	U. L.	10	2
プロバイダ A	サービス A2	B. E.		
プロバイダ B	サービス B1	U. L.	20	1
プロバイダ B	サービス B2	B. E.		
プロバイダ B	サービス B3	U. L.	10	3

【図 4】

(a)

開始バイト	長さ	タイプ	
0	32	文字列	サービスプロバイダ名称
32	32	文字列	サービス名称
64	-	-	可変部

901 902 903

(b)

サービスプロバイダ名称	サービス名称	テーブルインデクス
サービスプロバイダA	サービスA1	1
サービスプロバイダA	サービスA2	2
サービスプロバイダB	サービスB1	3

905 906 907

(c)

1	開始バイト	長さ	タイプ	
	64	4	整数	口座番号
	68	12	文字列	タイムスタンプ
	80	8	整数	引き出し額

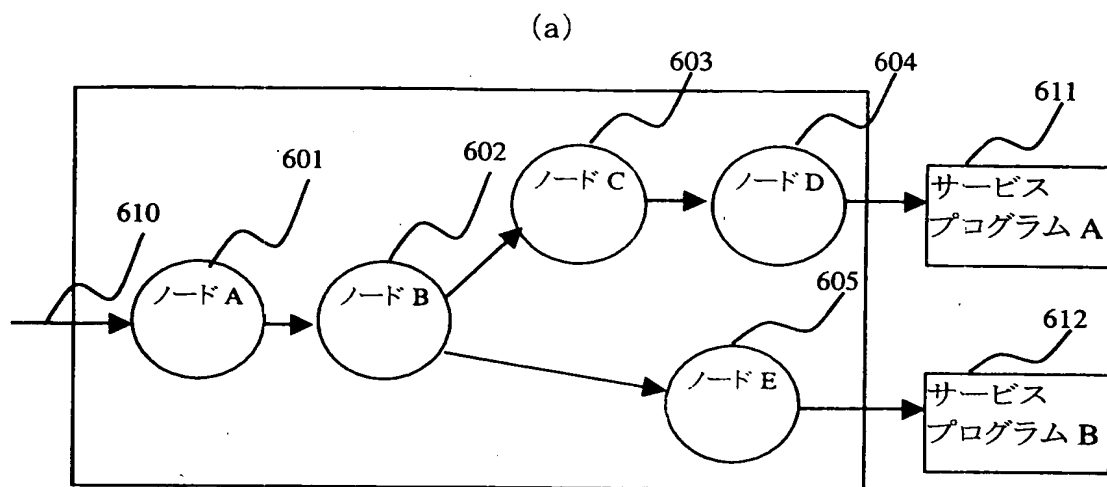
908 909 910 911

(d)

2	開始バイト	長さ	タイプ	
	64	4	整数	口座番号
	68	12	文字列	タイムスタンプ
	80	8	整数	現在残高

912 913 914 915

【図 5】



(b)

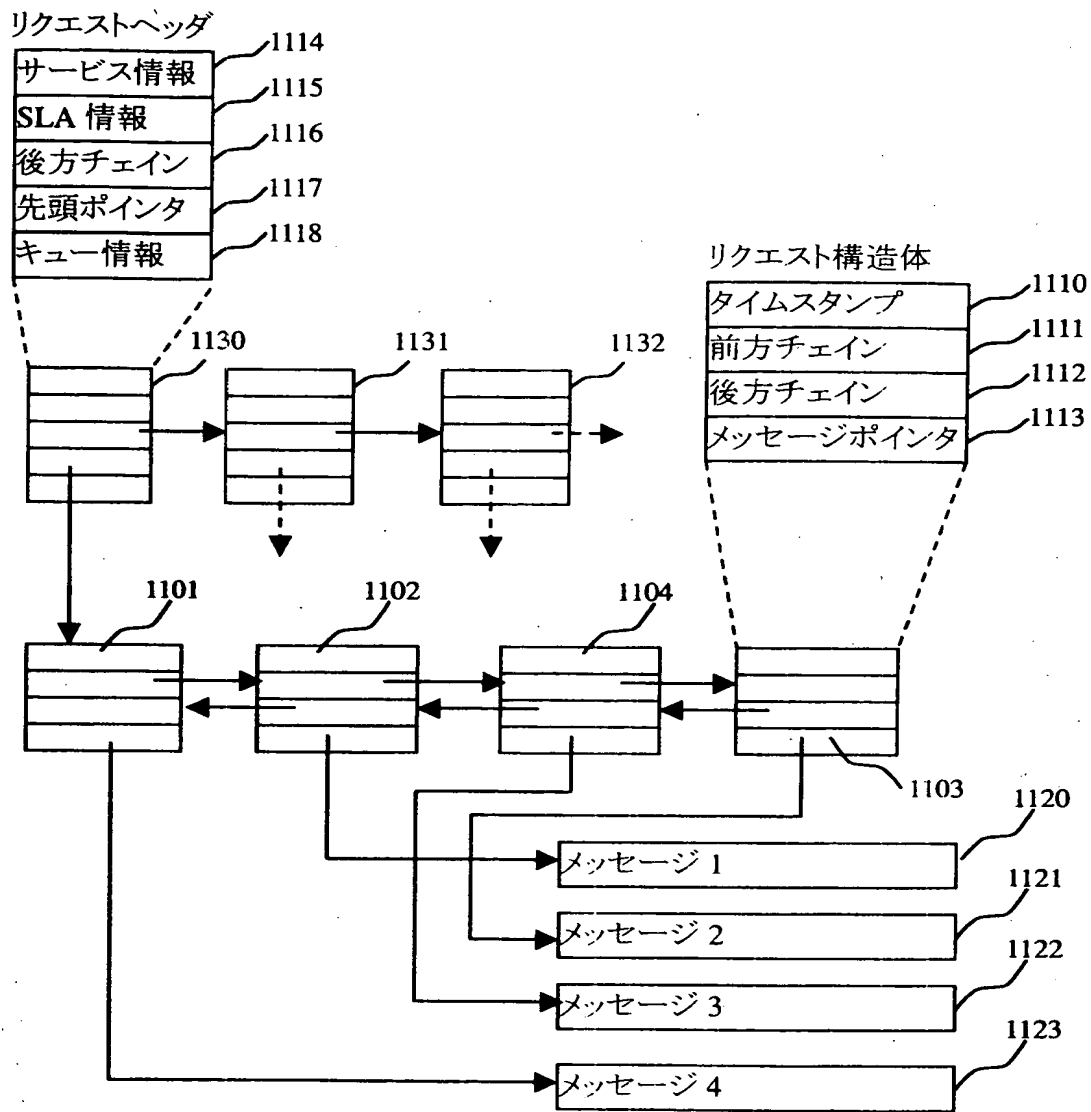
サービス名	サービスA	サービスA	サービスA	サービスA	サービスA
ノード名称	ノードA	ノードB	ノードC	ノードD	ノードE
ノード種類	入力ノード	処理ノード	変換ノード	出力ノード	出力ノード
入力先		ノードA	ノードB	ノードC	ノードB
出力先1	ノードB	ノードC	ノードD		
出力先2		ノードE			
モジュール	"IN1"	"AG"	"TR"	"O1"	"O2"

【図 6】

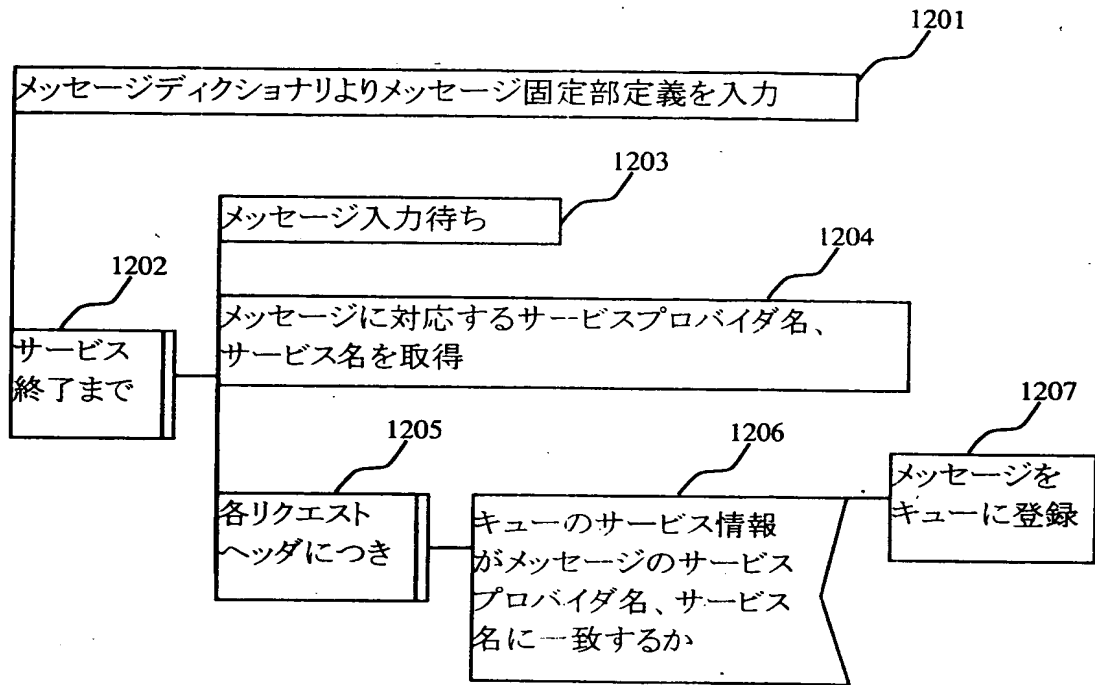
サービス プロバイダ 名称	サービス 名称	可変部		
		口座番号	タイム スタンプ	引き出し額

1001 1002 1003 1005 1006 1007

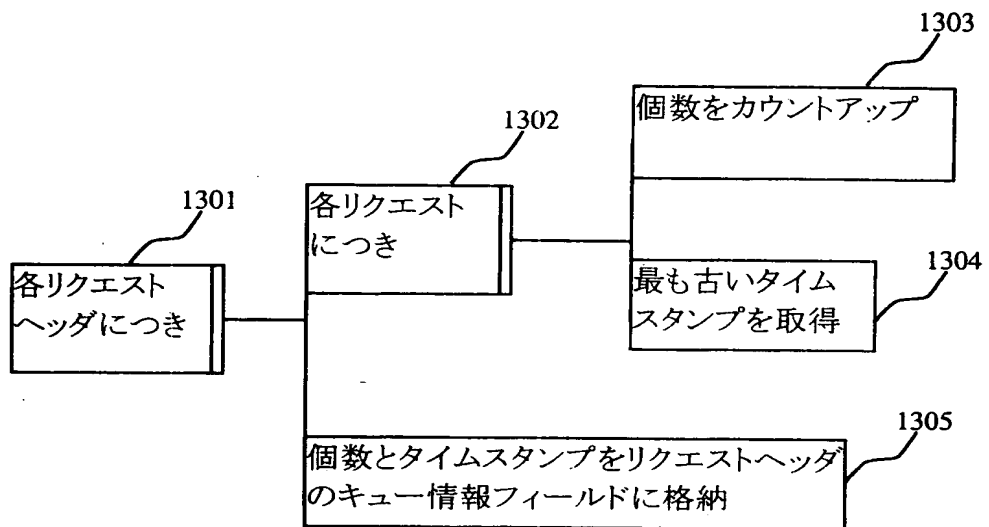
【図 7】



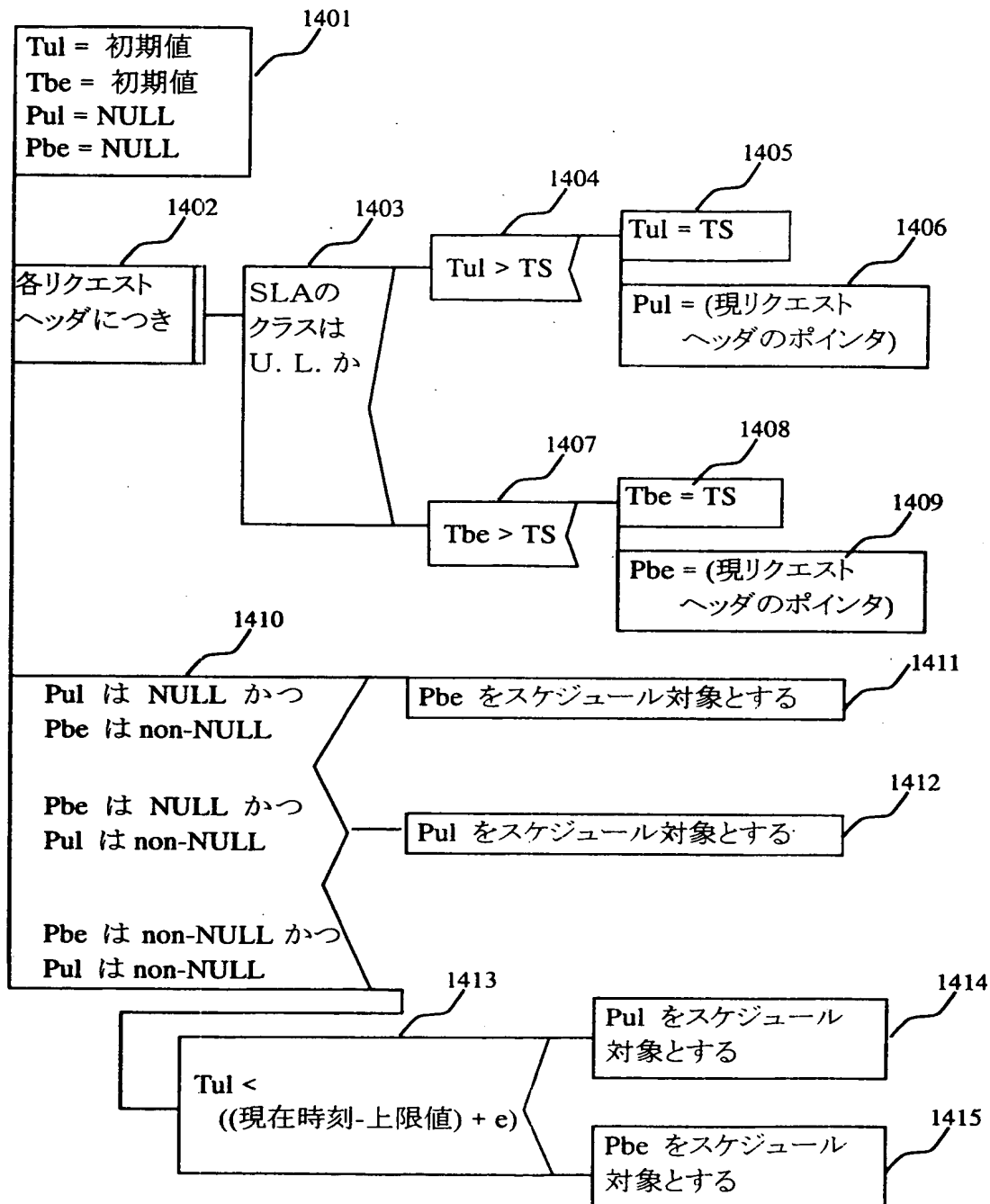
【図 8】



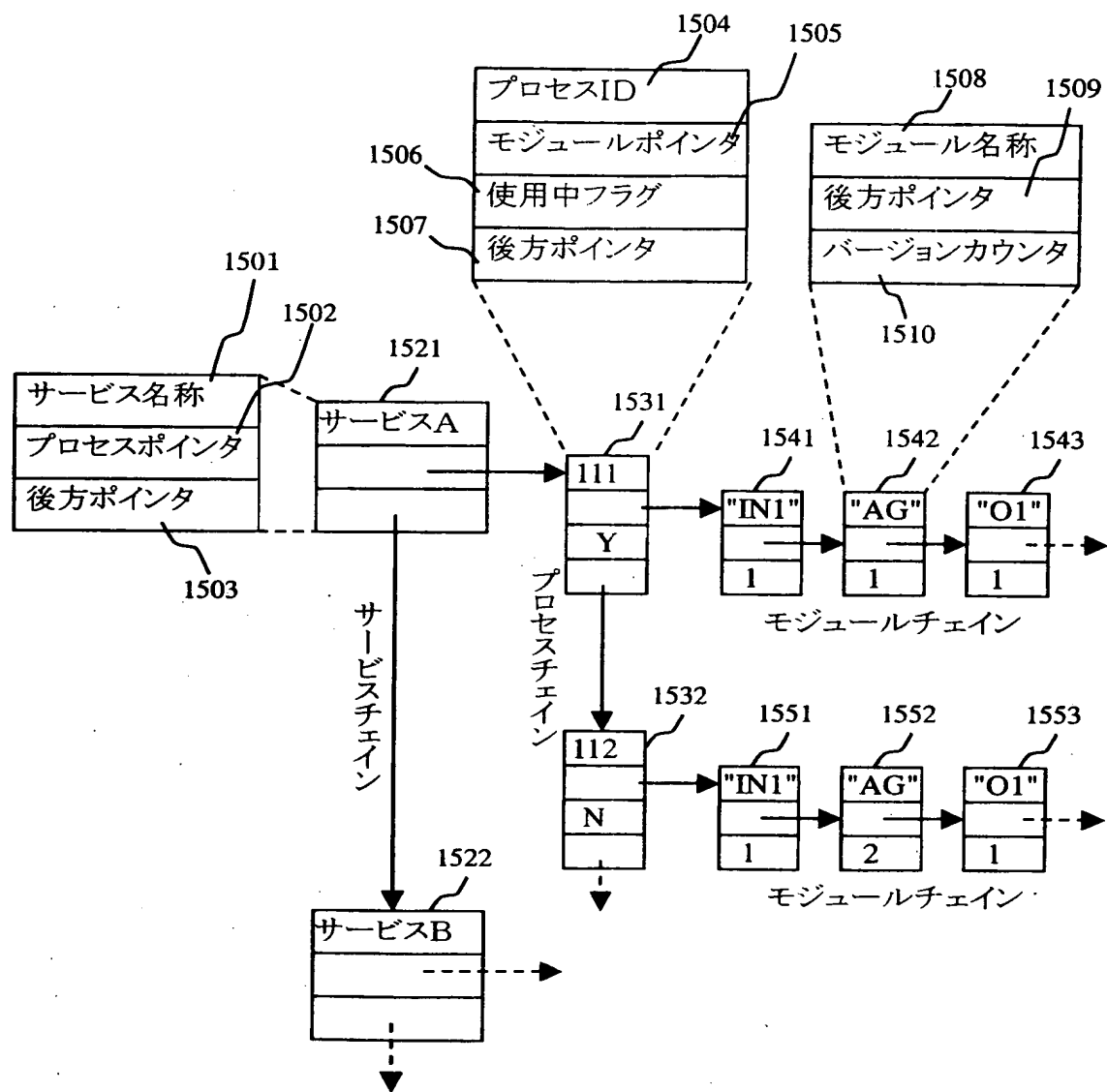
【図 9】



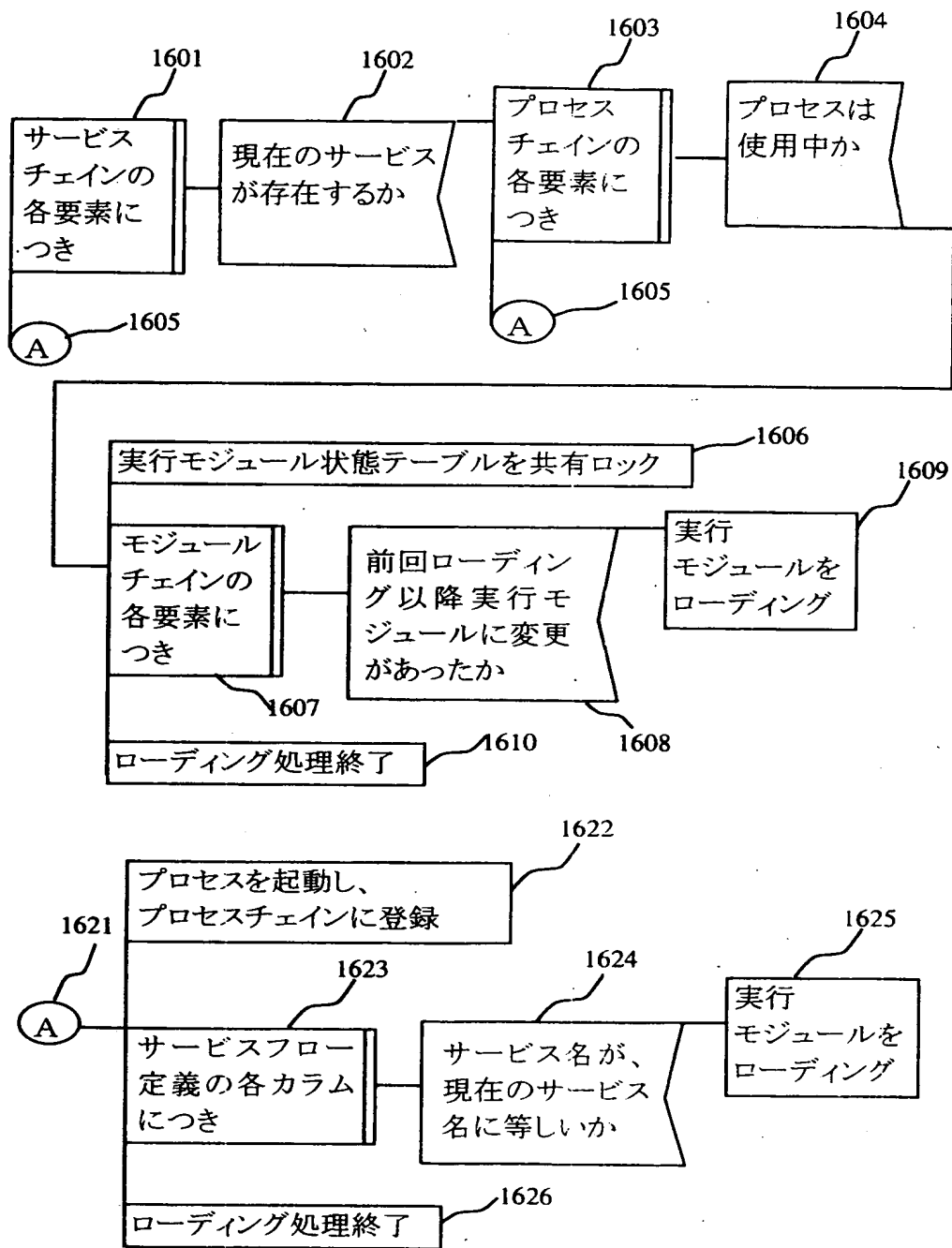
【図 1 0】



【図 11】



【図 12】



【図 1 3】

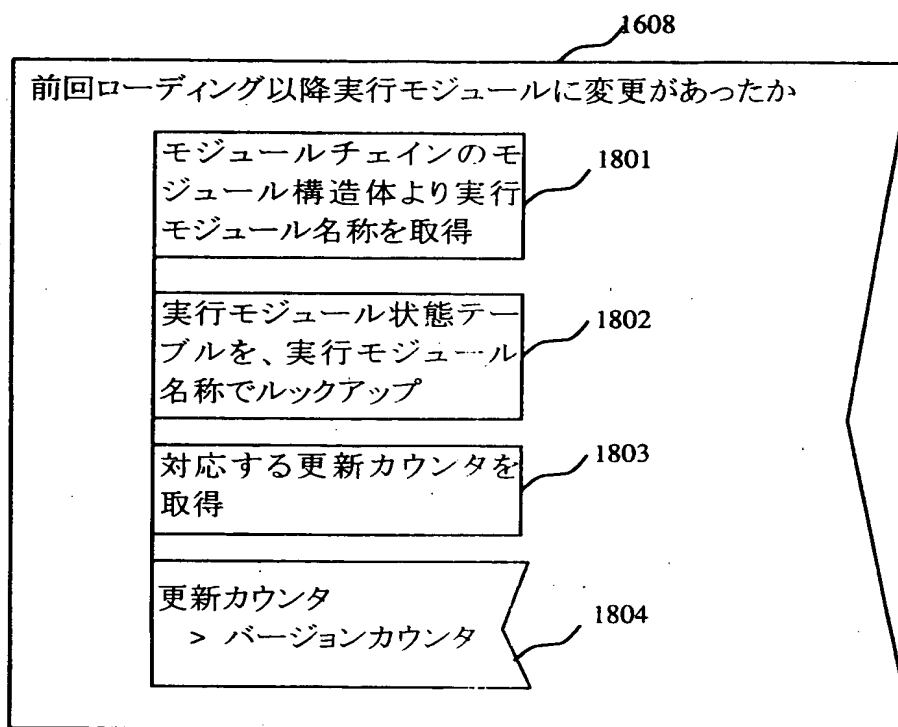
モジュール名称	"IN1"	"AG"	"TR"	"O1"	"O2"
更新カウンタ	1	1	2	3	1

1700 1701 1702 1703 1704 1705

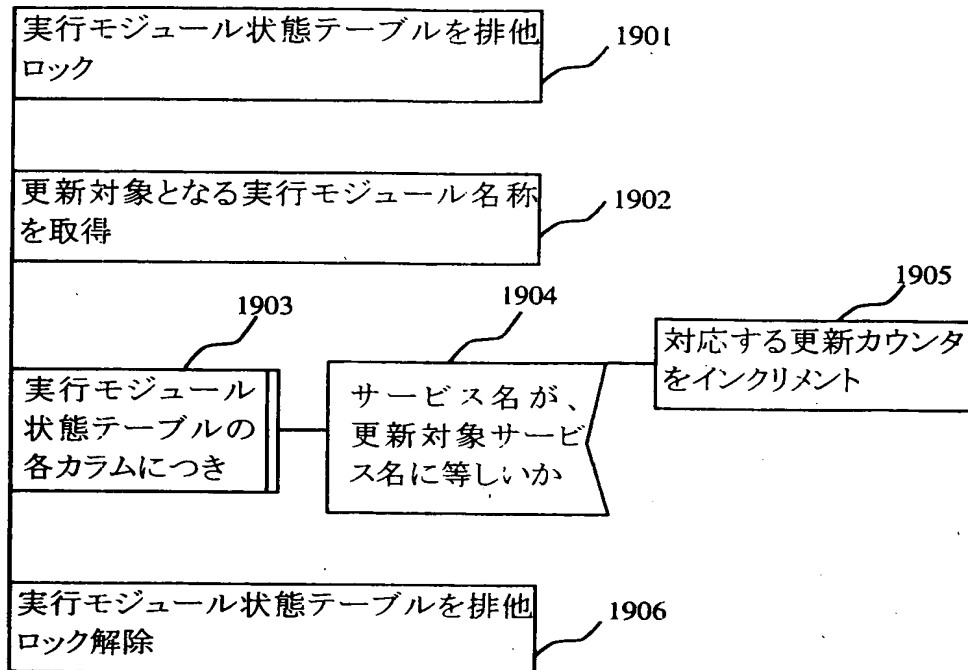
ロックフィールド	N
----------	---

1710

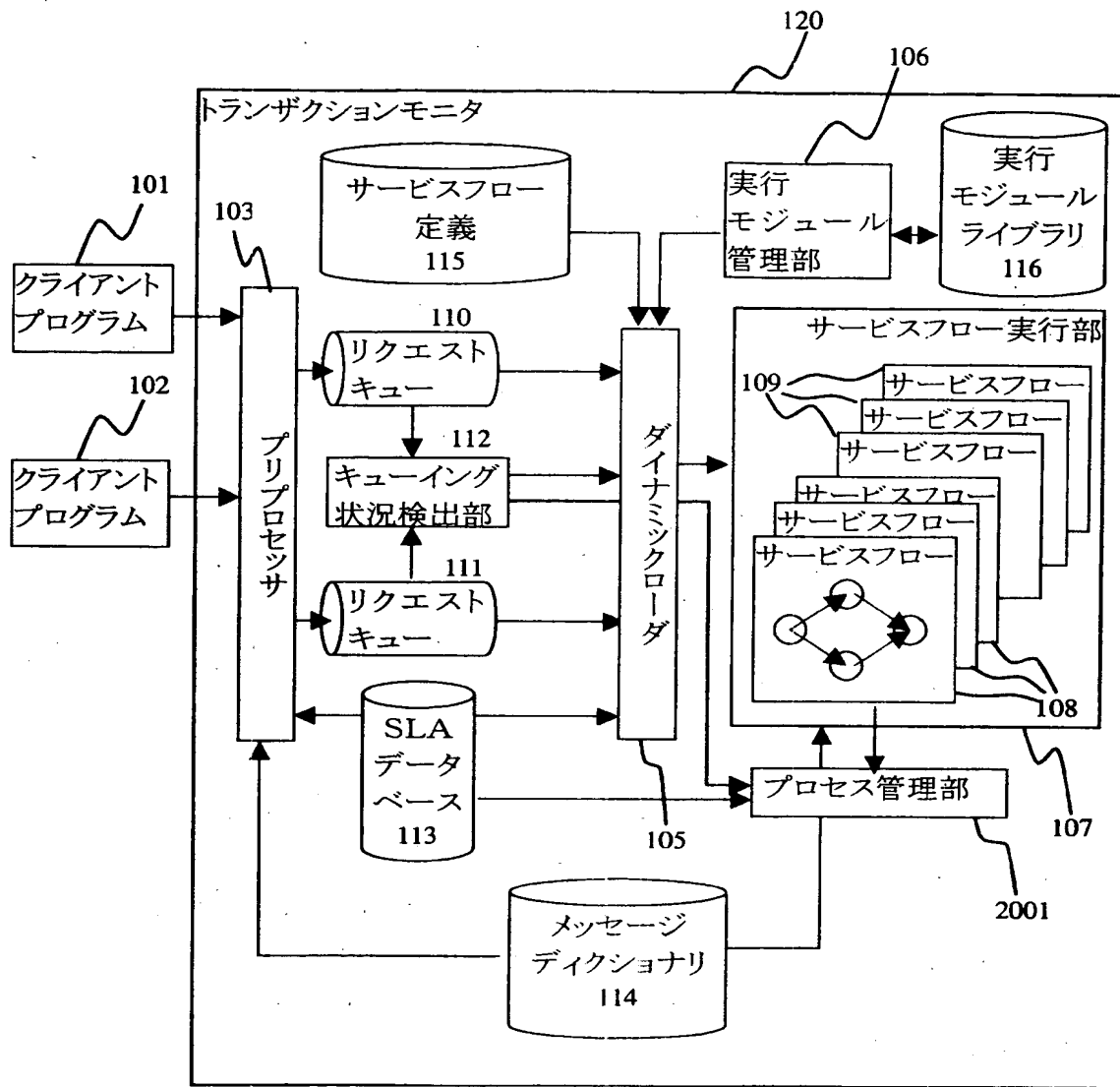
【図14】



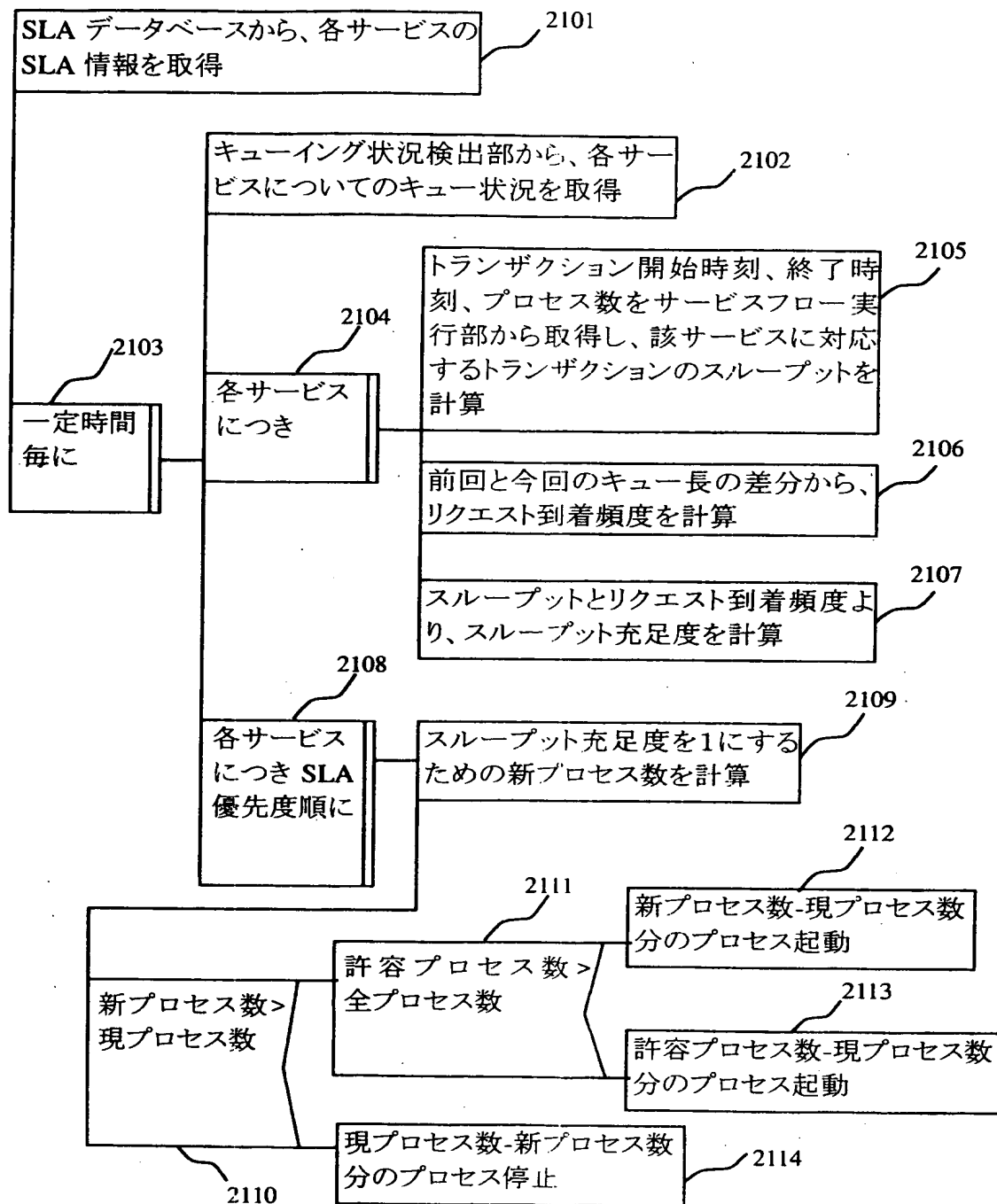
【図 15】



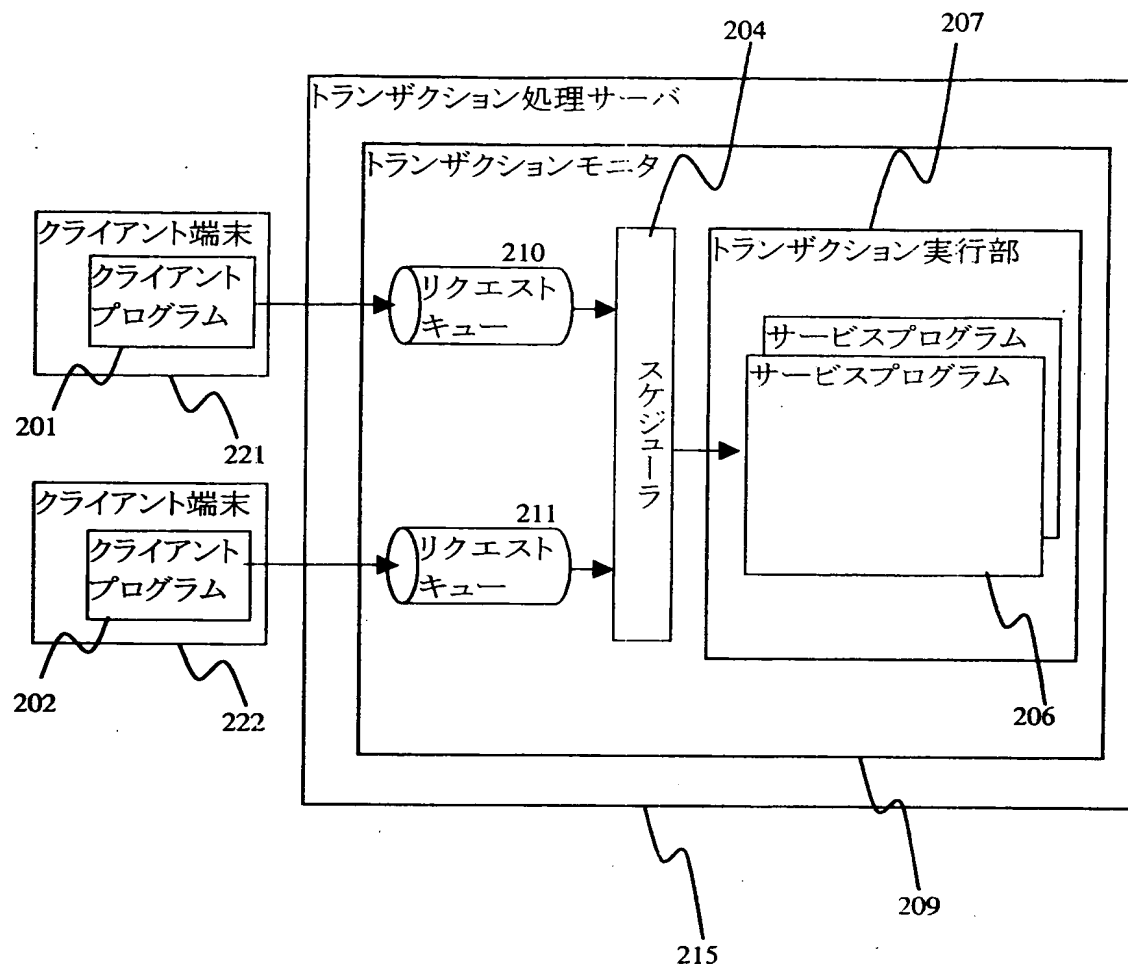
【図 16】



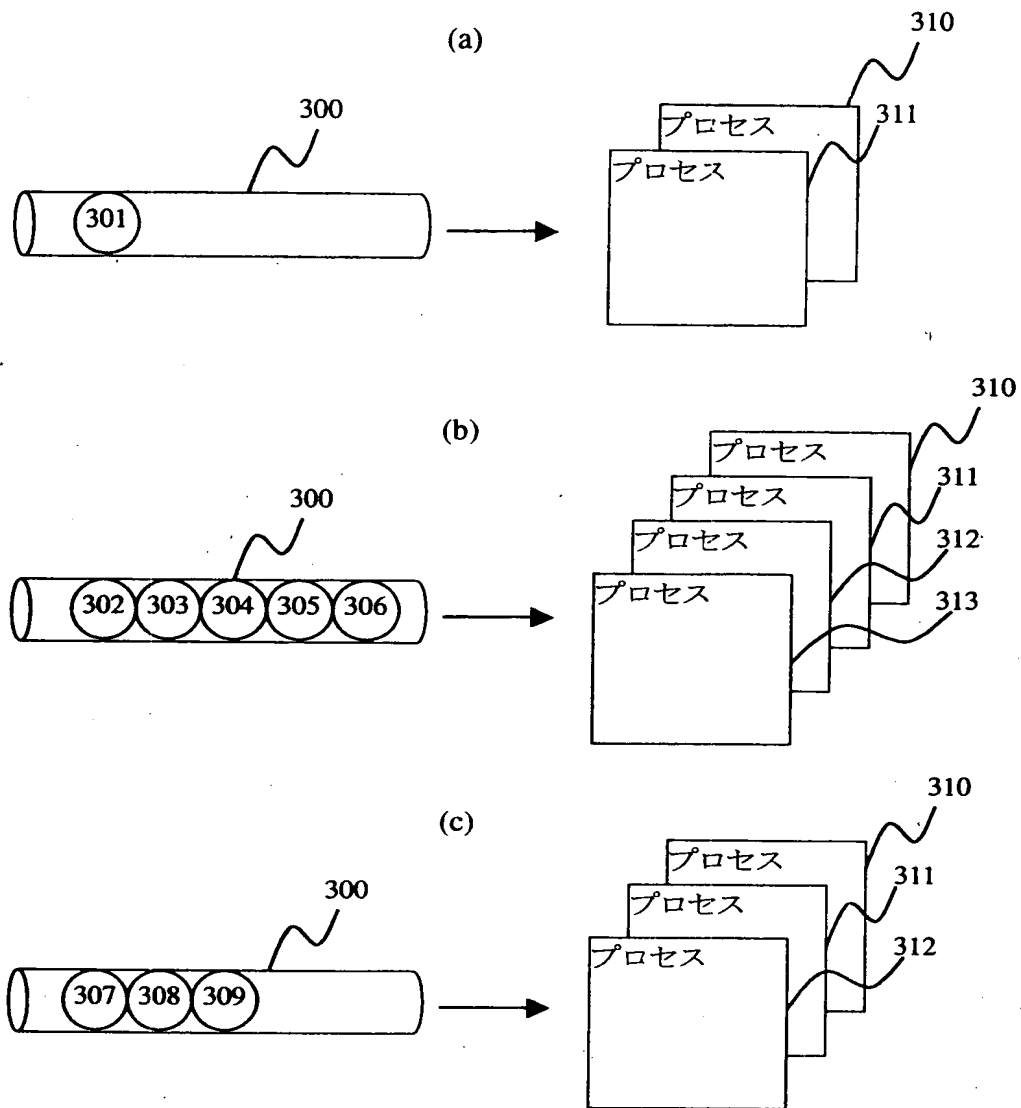
【図 17】



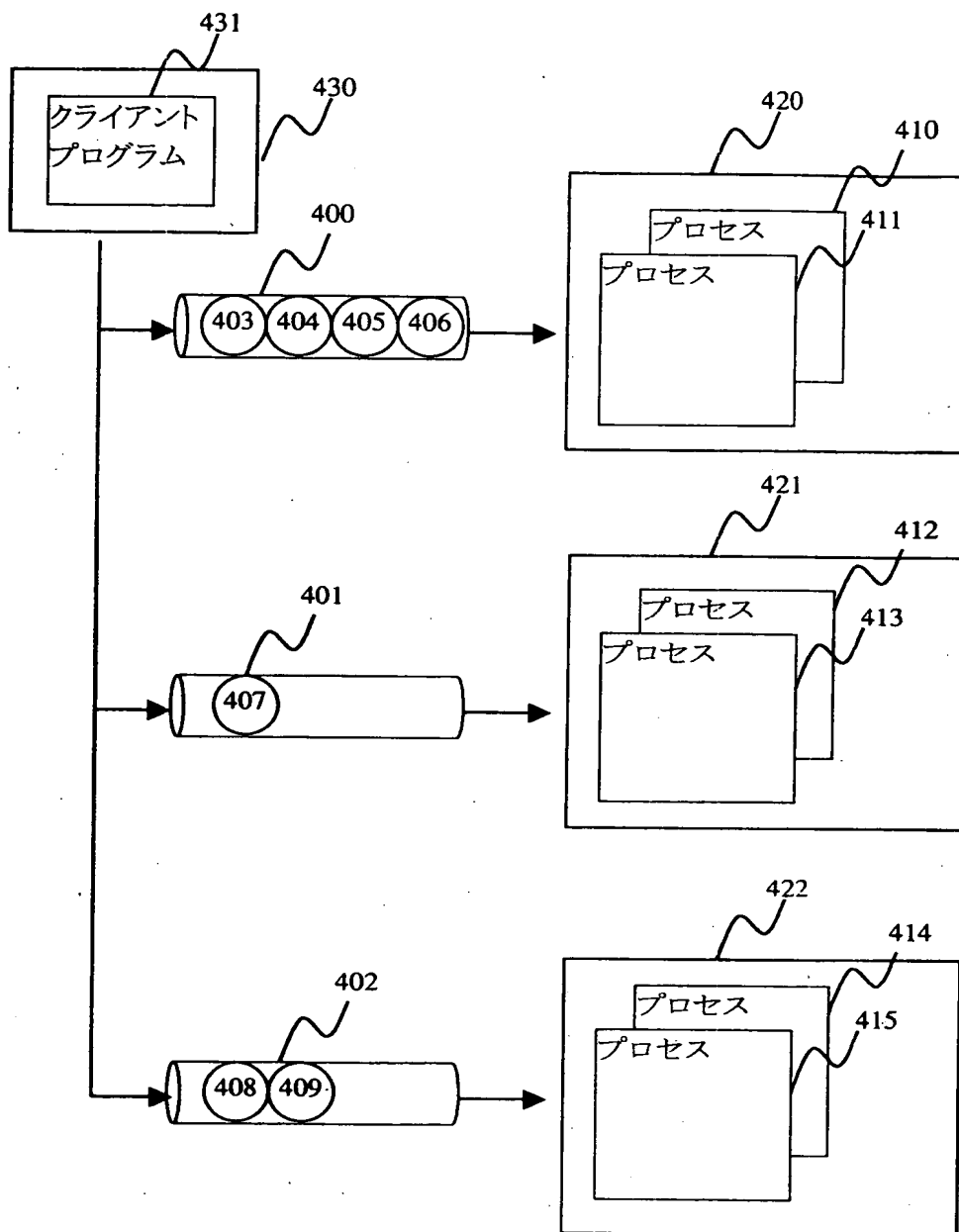
【図 1 8】



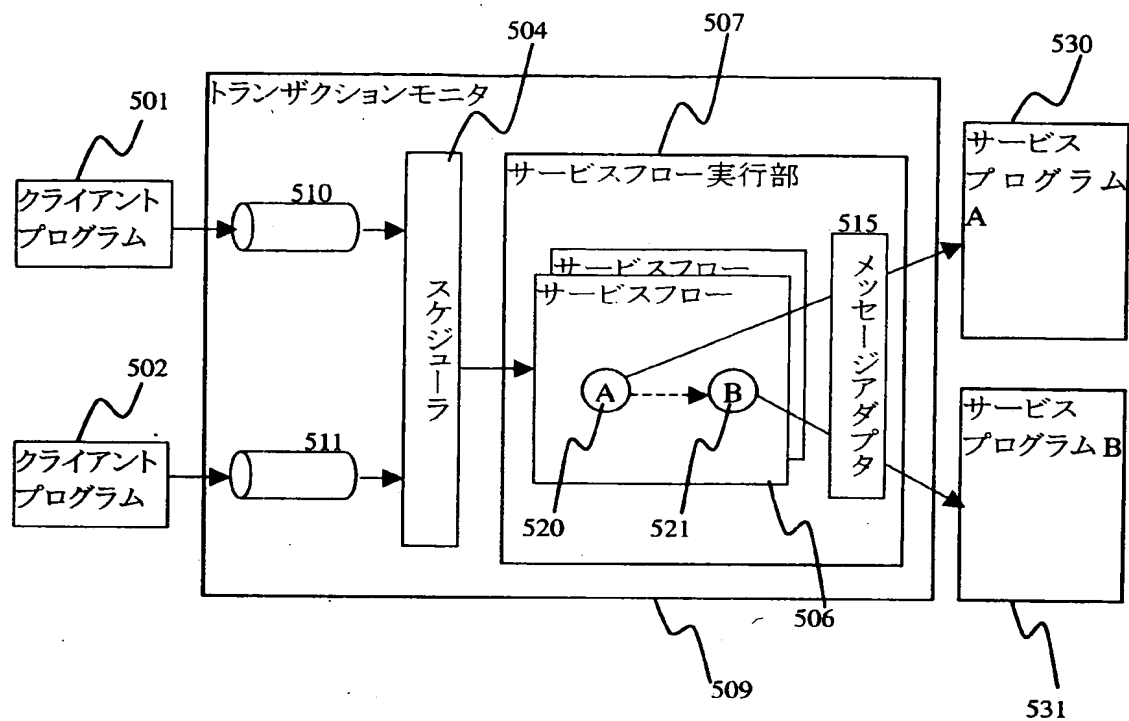
【図 1 9】



【図 20】



【図 21】



【書類名】 要約書

【要約】

【課題】 トランザクション処理システムにおいて、サービスレベル契約に従って複数のサービスを提供する。

【解決手段】 トランザクション処理システムが提供するサービスに対応して定義された契約情報を格納するSLAデータベース113と、クライアントから提供するサービスに対して送付された処理要求を、提供するサービス毎に順序付けて格納するリクエストキュー110、111と、これらキューイング手段に格納した処理要求の滞留情報を取得するキューイング状況検出部112と、クライアントからトランザクション処理システムに投入された処理要求の優先度を、契約情報および処理要求の滞留情報を参照して決定するスケジューラ104とを有する。

【選択図】 図1

出 願 人 履 歷 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都千代田区神田駿河台4丁目6番地
氏 名 株式会社日立製作所